Problem Specification of *The Light Control Case Study* of the Journal of Universal Computer Science

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

6th September 2000

Contents

Introduction

This document contains a problem specification of the *Light Control Case Study* of the *Journal of Universal Computer Science*. This projection includes all formal and natural language specifications as well as all references within the document.

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

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

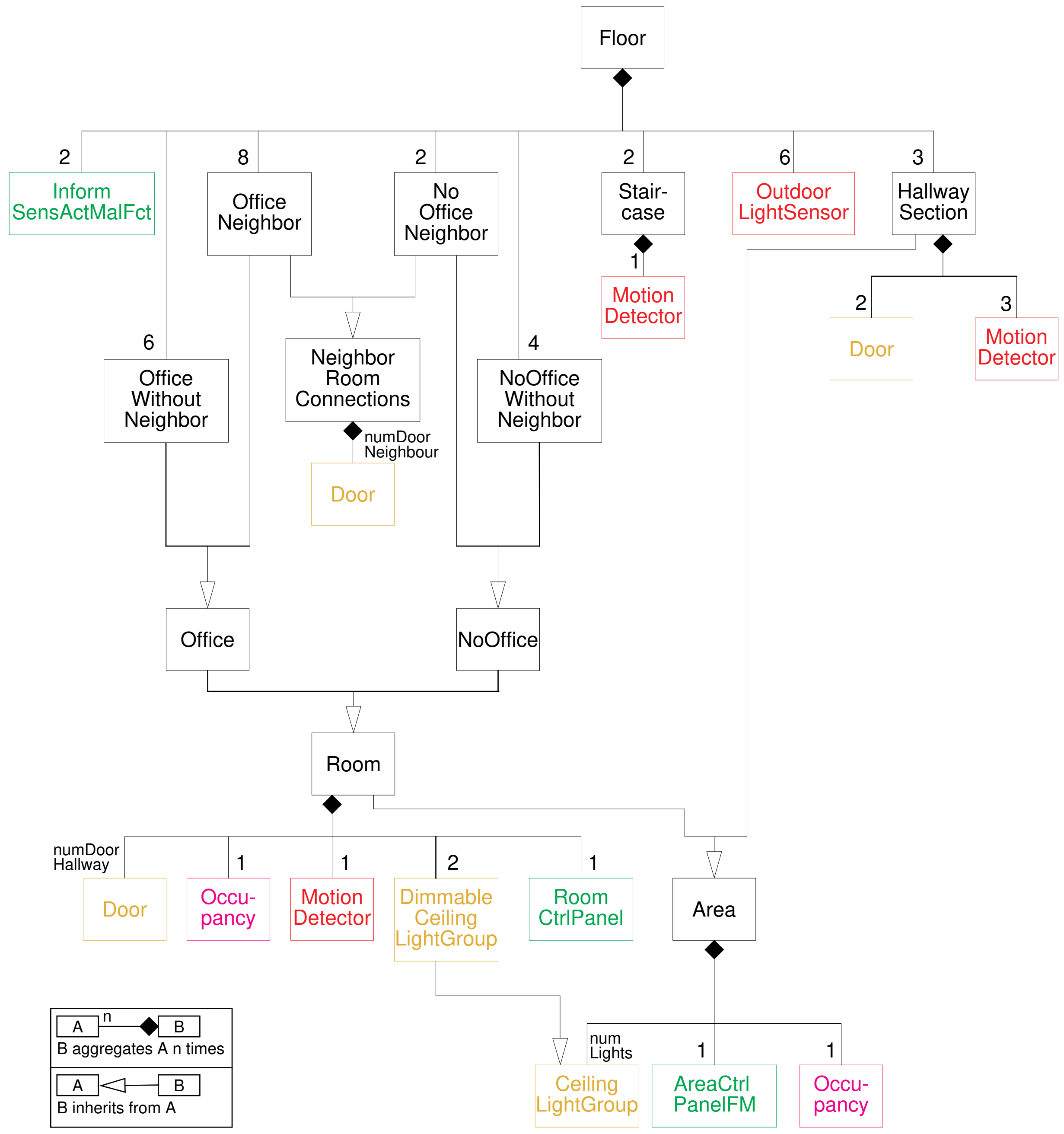

Figure 1: Class structure of group *BuildingUnits*

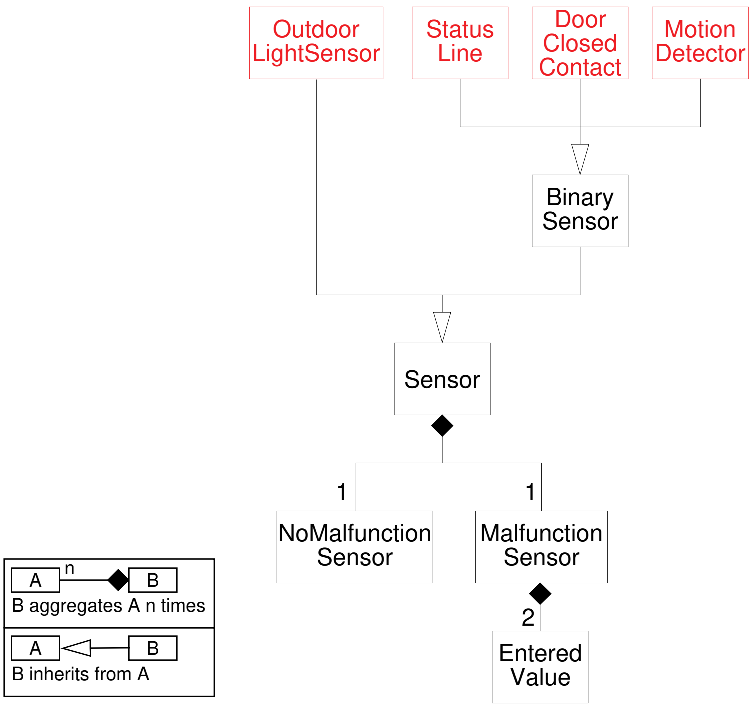

Figure 2: Class structure of group *Sensors*

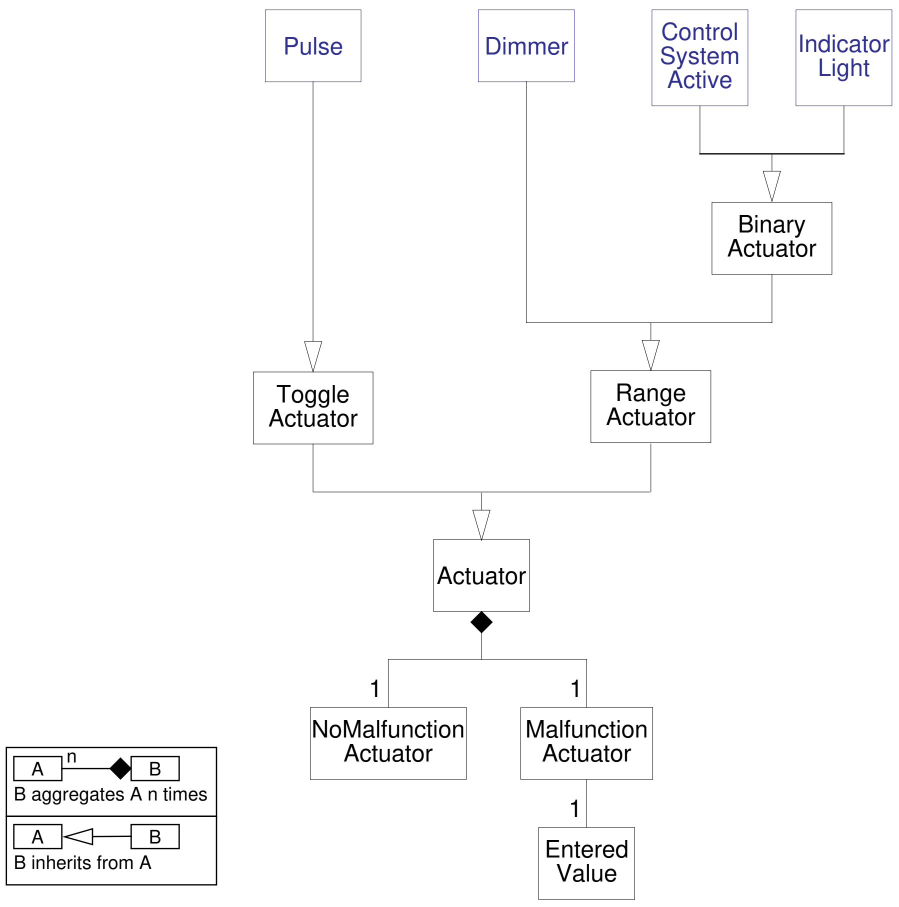

Figure 3: Class structure of group *Actuators*

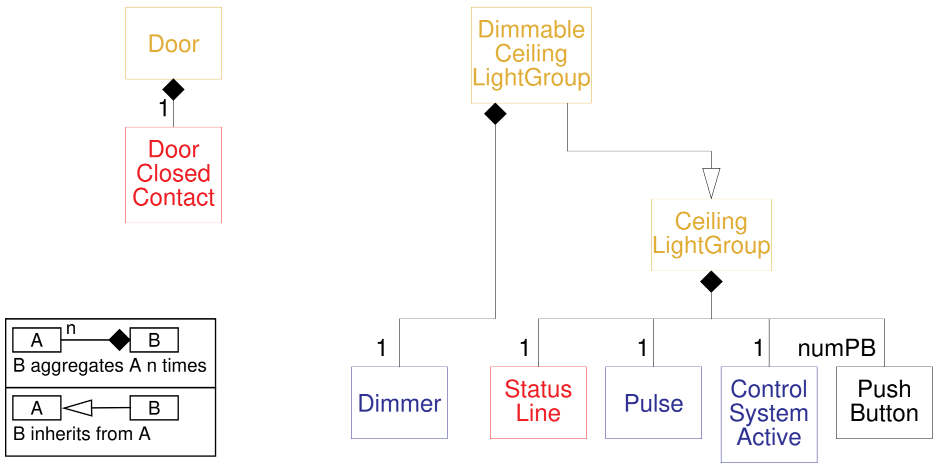

Figure 4: Class structure of group *CombinedSensorsAndActuators*

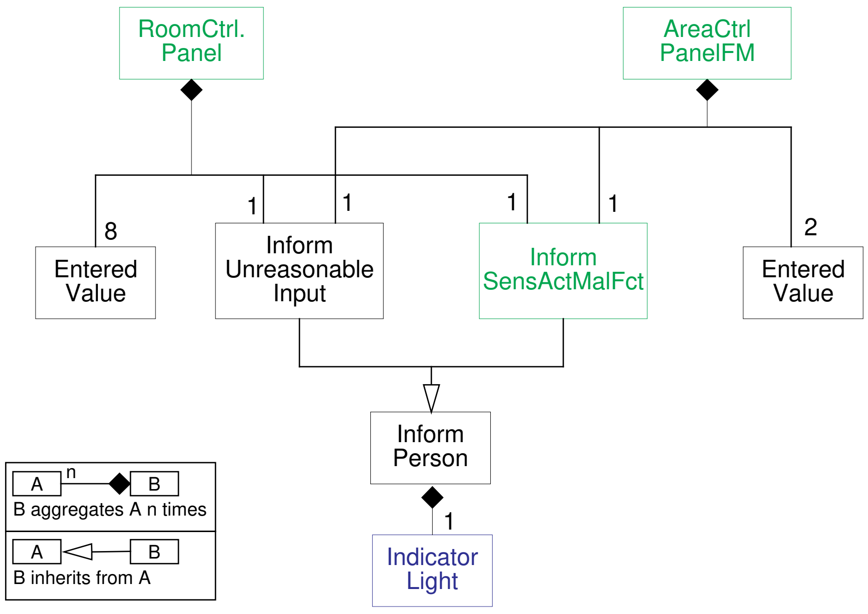

Figure 5: Class structure of group *HumanMachineInterface*

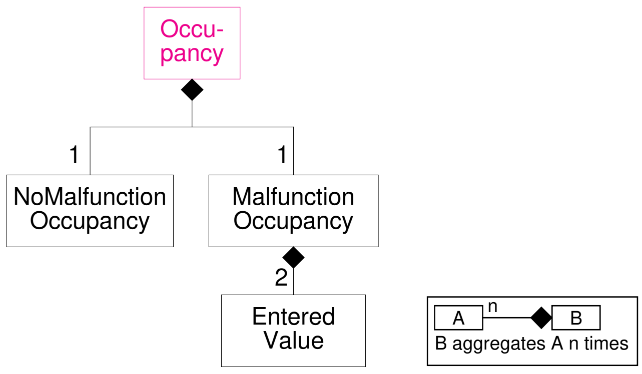

Figure 6: Class structure of group *Occupancy*

The Light Control Case Study: *Problem Description*

Stefan Queins
Gerhard Zimmermann
Martin Becker
Martin Kronenburg
Christian Peper
Rolf Merz
Jrgen Schfer

(University of Kaiserslautern, Germany
{queins, zimmerma, mbecker, kronburg, peper}@informatik.uni-kl.de,
merz@eit.uni-kl.de)

Abstract: This document contains a range of needs and requirements concerning the construction of a light control system for a floor of a university building. A description of the building architecture and of some pre-installed (light-)hardware is included. This problem description was the common input for all participants of the requirements engineering case study "Light Control".

Key Words: requirements engineering, building automation, problem description

Introductory Note

This document gives an informal description of the problem "Light Control System", that is the subject of the considered case study. It is based on two previous versions that have been used in the Sonderforschungsbereich 501 "Development of Large Systems with Generic Methods", a large project at the Computer Science Department of the University of Kaiserslautern. The initial version was created in 1995 by Stefan Queins and Gerhard Zimmermann. The second version (reported in [Fe+99]) integrates several changes by Martin Becker and Martin Kronenburg. This version was also used in a Dagstuhl Seminar on "Requirements Capture, Documentation, and Validation" that took place in June 1999.

Finally, the version presented here is the result of repeated revisions by Martin Kronenburg and Christian Peper in agreement with Rolf Merz and Jrgen Schfer from the Electrical Engineering Department of the University of Kaiserslautern, who were acting as customers in the light control case study (LCCS). Some additional improvements are based on input received from Daniel Berry, Vincenzo Gervasi, Julio Leite, and Vinicius da Silva Almendra.

Thus, the basic intention of providing a customer document as the basis for the case study has been preserved. The revisions were intended to reduce the need for customer feedback during the LCCS. Furthermore, to achieve solutions that are better comparable, all interactions between the customer and the participants of the case study have been published on the web [CF99].

The problem description "Light Control" is divided into 4 parts. [Part 1] is a brief introduction. [Part 2] describes the architecture of the 4th floor of a university building in Kaiserslautern, which is the subject of the informal needs given in [Part 3]. Finally, [Part 4] lists and explains technical terms that are used in the document.

[Fe+99] R. L. Feldmann, J. Mnch, S. Queins, S. Vorwieger, G. Zimmermann: Baselining a Domain-Specific Software Development Process, Technical Report SFB501 TR-02/99, University of Kaiserslautern, 1999

[CF99] Customer Feedback of the LCCS, http://rn.informatik.uni-kl.de/~recs/qna/, University of Kaiserslautern, 1999

Part 1: Introduction

1 The main motivation for the development of a new light control system are the disadvantages of the currently existing system. Since all lights are controlled manually, electrical energy is wasted by lighting rooms which are not occupied and by little possibilities to adjust light sources relative to need and daylight.

2 In the following document, ***keywords*** are marked at their first occurrence and listed in the additional dictionary (Part 4).

3 Words written in *emphasis* are names of physical sensors/actuators.

4 Paragraphs are numbered for easier reference.

Application Knowledge

Part 2: Floor Description

Fact *BD_5*

Information : In this part, the architecture and the installation of the given sensors and actuators of Building 32, 4th floor is described

Fact *Figure1*

Information :

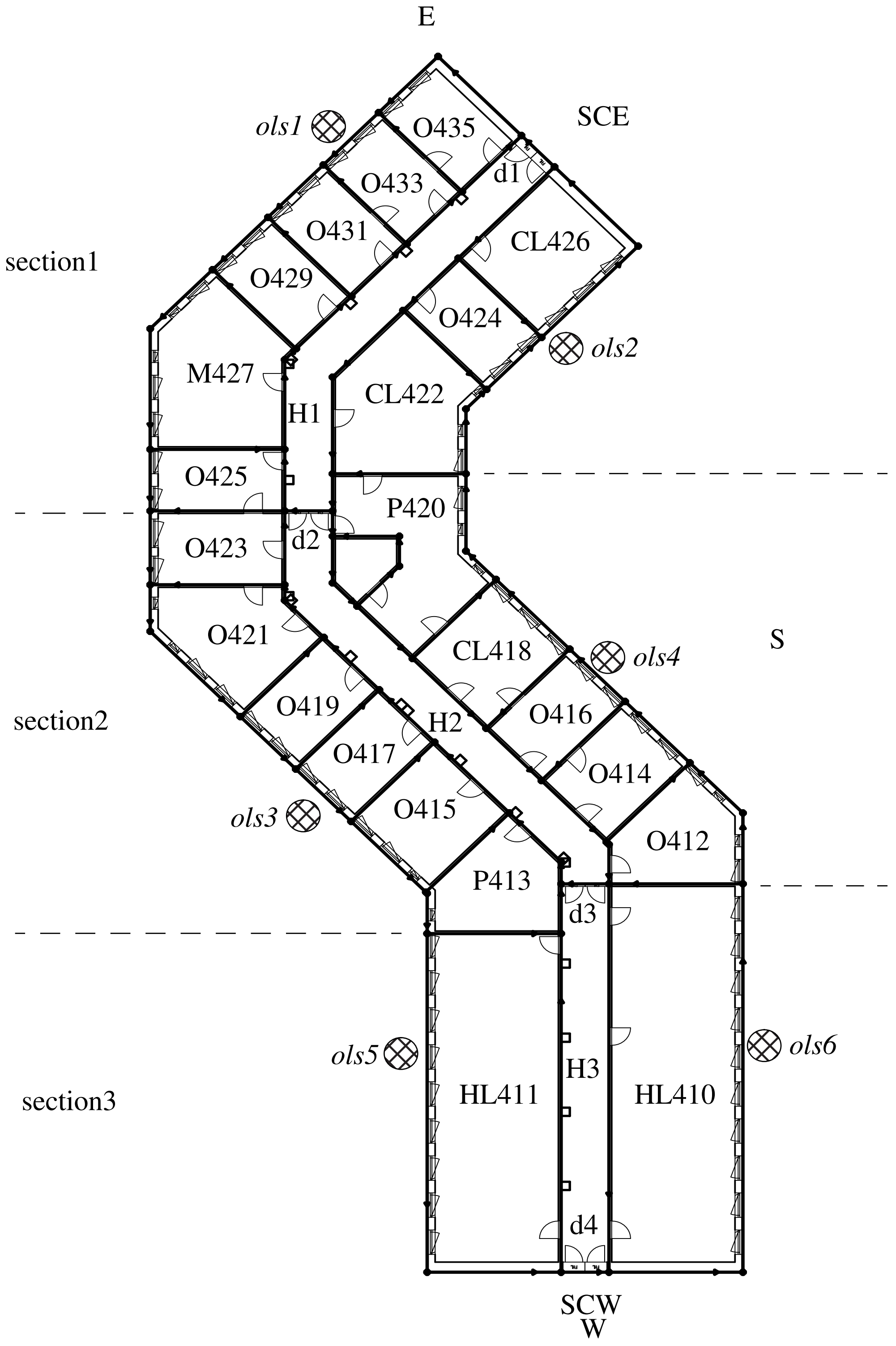

Figure 1: Architecture of the 4th Floor of Building 32

Postreferences : lsref Floor : o415
lsref Floor : o417
lsref Floor : o419
lsref Floor : o429
lsref Floor : o425
lsref Floor : o431
lsref Floor : o435
lsref Floor : o423
lsref Floor : o433
lsref Floor : o421
lsref Floor : m427
lsref Floor : hl411
lsref Floor : cl426
lsref Floor : o424
lsref Floor : cl422
lsref Floor : cl418
lsref Floor : o416
lsref Floor : o414
lsref Floor : o412
lsref Floor : hl410
lsref Floor : h1
lsref Floor : h2
lsref Floor : h3
lsref Floor : scw
lsref Floor : sce
lsref Floor : ols1
lsref Floor : ols2
lsref Floor : ols3
lsref Floor : ols4
lsref Floor : ols5
lsref Floor : ols6
lsref Floor : ROOMS
lsref Floor : HALLWAYS
dkref Floor : D_F1
dkref Floor : D_F2
dkref Floor : D_F3
dkref Floor : D_F4
dkref Floor : D_F5
dkref Floor : D_F6
dkref Floor : D_F7
dkref Floor : D_F8

Fact *BD_6*

Information : The fourth floor of Building 32 consists of three sections and shares two staircases, staircase east (SCE) and staircase west (SCW), with other floors of the building, as shown in Figure 1. Each section is divided into some hallway sections (H) and rooms, each of which may be an office (O), a computer lab (CL), a hardware lab (HL), a peripheral room (P), or a meeting room (M). All rooms in a section are accessible via a connected hallway section. There are three hallway sections and 22 rooms to control. Figure 1 shows also the six outdoor light sensors (ols1 - ols6) and the major compass directions. The sensors cover the six directions of the different walls. The label in a room indicates the type of the room and gives a unique number, see Figure 1.

Postreferences : lsref Floor : h1
lsref Floor : h2
lsref Floor : h3
lsref Floor : scw
lsref Floor : sce
lsref Floor : ols1
lsref Floor : ols2
lsref Floor : ols3
lsref Floor : ols4
lsref Floor : ols5
lsref Floor : ols6
lsref Floor : ROOMS
lsref Floor : HALLWAYS

2.1 Office Description

Fact *BD_7*

Information : Each office (shown in Figure 2) has one door (d1) to the hallway section and can have up to two doors (d2, d3) leading to its adjacent rooms. Each door is equipped with a door closed contact, named dcc<n>, where n is the number of the door in the room.

Postreferences : lsref Door : doorClosedContact
lsref Room : hallwayDoor
lsref NeighborRoomConnections : neighborDoor
dkref NeighborRoomConnections : D_NRC1

Fact *BD_8*

Information : Each office is equipped with

1. one motion detector (imd), so that the room is fully covered.
2. two ceiling light groups (window and wall). The luminaries in a ceiling light group in any room are turned on or off only as a group.
   Each ceiling light group is controlled by one push button on the wall (pb1 and pb2, respectively), which toggles the ceiling light group if pushed.
   A ceiling light group in a room shows the following behavior if the corresponding push button is pushed:
   (i) if the ceiling light group is completely on, it will be switched off
   (ii) otherwise it will be switched on completely.
3. Each ceiling light group can be dimmed with its own dimmer-actuator.
4. two status lines (sll1 and sll2) each of which shows the status of one ceiling light group.

Postreferences : dkref PushButton : D_PB1
dkref PushButton : D_PB2
lsref LightScene : CEIL_LIGHT_GROUPS
lsref PushButton : T_PB
lsref PushButton : light
lsref PushButton : pushBut
lsref CeilingLightGroup : sl
lsref CeilingLightGroup : pb
lsref Area : light
lsref Room : md

Fact *Figure2*

Information :

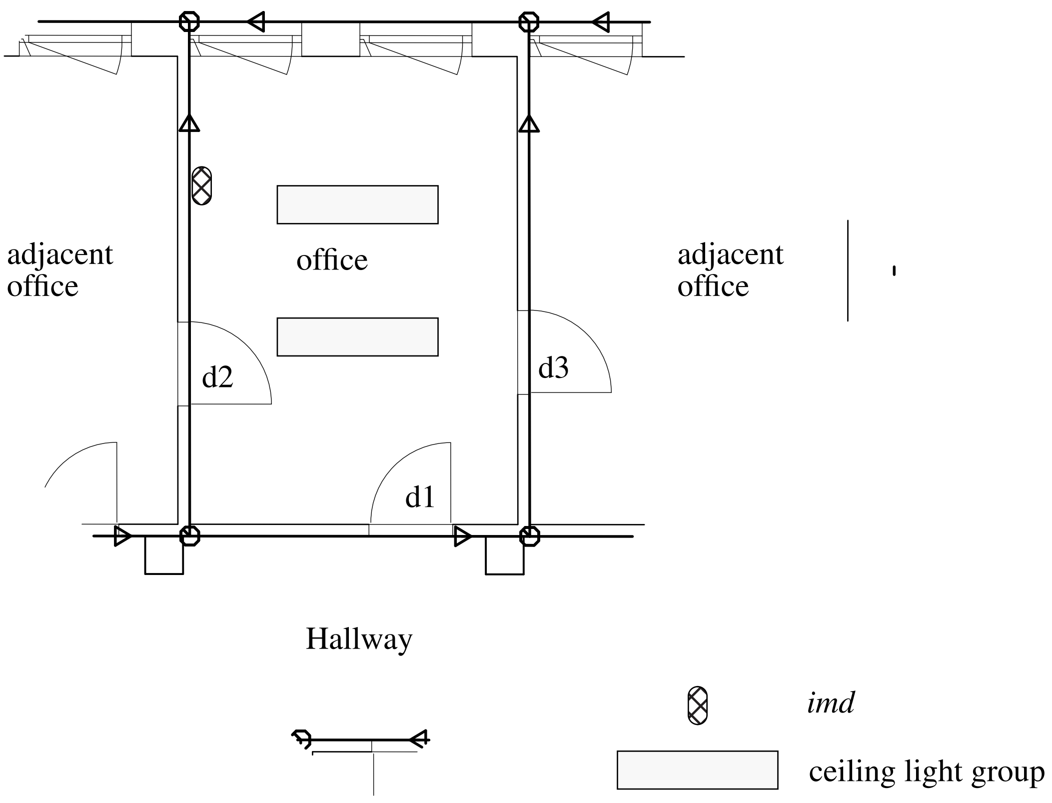

Figure 2: Office Architecture

2.2 Computer Lab Description

Fact *BD_9*

Information : Same as office.
Postreferences : lsref Room : md
lsref Room : hallwayDoor

2.3 Hardware Lab Description

Fact *BD_10*

Information : Same as office, but with more than one door leading to the hallway section.
Postreferences : lsref Room : md
lsref Room : hallwayDoor

2.4 Meeting Room Description

Fact *BD_11*

Information : Same as office.
Postreferences : lsref Room : md
lsref Room : hallwayDoor

2.5 Peripheral Room Description

Fact *BD_12*

Information : The peripheral rooms will not be controlled by the control system, and thus they are not described here!

2.6 Hallway Section Description

Fact *BD_13*

Information : Each hallway section is limited by two doors, each of which is leading to an adjacent hallway section or to an adjacent staircase. The associated names of the doors (d1, d2, d3, d4) are shown in Figure 1. Each door is equipped with a door closed contact, named dcc<n>, where n is derived from the label of the door.

Postreferences : lsref DimmableCeilingLightGroup : dimmer
lsref Door : doorClosedContact
lsref HallwaySection : eastDoor
lsref HallwaySection : westDoor

Fact *BD_14*

Information : Each hallway section is equipped with

1. two motion detectors (imd1 and imd2), placed above the doors at each end of the hallway section to determine the presence of a person near a door,

2. one motion detector to cover the whole section (imd3),

3. one hallway section ceiling light group. The luminaries in a hallway section ceiling light group are turned on or off only as a group.
   Each ceiling light group is controlled by several push buttons (pb<i>) each of which toggles the ceiling light group if pushed. A hallway section ceiling light group shows the following behavior if a push button is pushed:

   (i) if the hallway section ceiling light group is on, then it will be switched off

   (ii) otherwise it will be switched on

4. one status line (sll1) that shows the status of the hallway section ceiling light group.

Postreferences : dkref PushButton : D_PB1
dkref PushButton : D_PB2
lsref PushButton : T_PB
lsref PushButton : light
lsref PushButton : pushBut
lsref CeilingLightGroup : sl
lsref CeilingLightGroup : pb
lsref Area : light
lsref HallwaySection : eastMd
lsref HallwaySection : westMd
lsref HallwaySection : middleMd

2.7 Staircase Description

Fact *BD_15*

Information : Each staircase connects several floors.

Fact *BD_16*

Information : At the landing of each staircase at each floor, the staircase is equipped with

1. one motion detector (imd1) above the door of the landing that leads to the adjacent hallway section to detect motion in the staircase near the door.

Postreferences : lsref Staircase : md

2.8 Sensor Description

Fact *BD_17*

Information : Analog sensors typically have an exponential response time. Conversion time is the time to convert the analog value to a digital one that can be accessed by the control system. Reaction time is the time from a change of the sensed property to the time when the sensor has reached 90

Postreferences : lsref NoMalfunctionSensor : reactionTime
lsref NoMalfunctionSensor : conversionTime
lsref NoMalfunctionSensor : envEntity
lsref NoMalfunctionSensor : measuredEntity
lsref NoMalfunctionSensor : convertedEntity
lsref NoMalfunctionSensor : modifyReaction
lsref NoMalfunctionSensor : modifyConversion
dkref NoMalfunctionSensor : D_NMS1
dkref NoMalfunctionSensor : D_NMS2

Fact *Table1*

Information :

Table 1: Sensors

Type	Reso-lution	Range	Reaction Time	Conver-sion Time	Description
door closed contact		0, 1	10 ms		It is placed above the door and is 1 if the door is fully closed, 0 otherwise
motion detector		0, 1	1 s		If set to 1, a person is moving, even very slowly, in the range of the detector.
status line		0, 1	10 ms		Senses if the light voltage is turned on (1) or off (0).
outdoor light sensor	1 lux	1-10000 lux	10 ms	1 s	Mounted perpendicular to facade, measures the *illumination* of the facade for the calculation of light flow through a window.

Postreferences : lsref Door : doorClosedContact
gsref LIGHT_VALUES
gsref BINARY
lsref NoMalfunctionSensor : reactionTime
lsref NoMalfunctionSensor : conversionTime
lsref OutdoorLightSensor : OUT_LIGHT_VALUES
dkref OutdoorLightSensor : D_OLS1
dkref OutdoorLightSensor : D_OLS2
dkref BinarySensor : D_BS1
dkref StatusLine : D_SL1
dkref DoorClosedContact : D_DCC1
dkref MotionDetector : D_MD1
lsref CeilingLightGroup : sl
lsref CeilingLightGroup : pulse

2.9 Actuator Description

Fact *BD_18*

Information : Actuators have a linear response time. Reaction time is therefore defined as the time to change from 0 to 100

Postreferences : lsref NoMalfunctionActuator : reactionTime
lsref NoMalfunctionActuator : envEntity
lsref NoMalfunctionActuator : adjustedEntity
dkref RangeActuator : D_RA1

Fact *Table2*

Information :

Table 2: Actuators

Type	Range	Control	Reaction Time	Description
control system active		0, 1	10 ms	If the control system sends a 1 within every 60 s, the control system is still alive.
dimmer	0-100%		10 ms	Controls light between 0 (off) and 10-100% (on).
pulse	0, 1		10 ms	If the value changes from 0 to 1, the light changes from on to off or from off to on.
push button		0, 1	10 ms	1 as long as pushed

Postreferences : gsref BINARY
gsref PERCENT
lsref NoMalfunctionActuator : reactionTime
dkref ToggleActuator : D_TA1
dkref ToggleActuator : D_TA2
dkref Pulse : D_P1
dkref Dimmer : D_D1
dkref ControlSystemActive : D_CSA1
lsref CeilingLightGroup : csa

2.10 Dimmable Light

Fact *BD_19*

Information : The structure of a dimmable light is shown in Figure 3. Inputs to a dimmable light are created by a pulse to toggle the light, by a dimmer to set the current dim value, and by control system active to show the status of the control system. If this signal is not sent every 60 s, the dimmable light switches to fail safe mode, i.e. dim value is assumed to be 100

Postreferences : lsref CeilingLightGroup : sl
lsref CeilingLightGroup : pulse
lsref CeilingLightGroup : pb
lsref DimmableCeilingLightGroup : dimmer
lsref Area : light
lsref CeilingLightGroup : csa

Fact *Figure3*

Information :

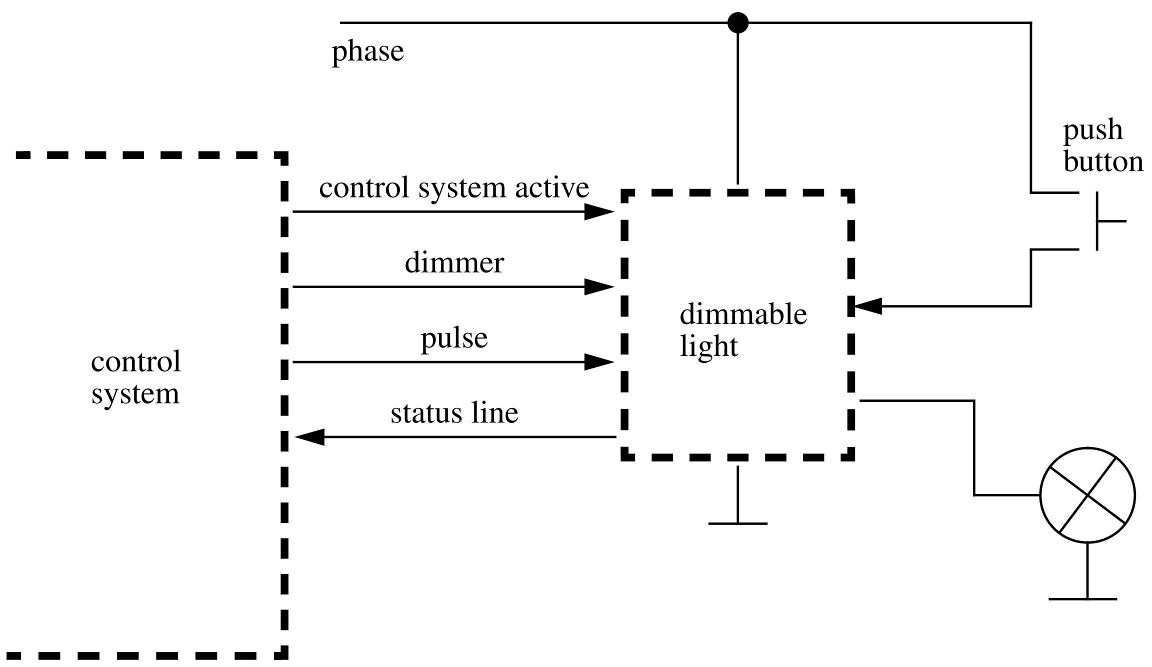

Figure 3: Dimmable Light

Postreferences : lsref CeilingLightGroup : sl
lsref CeilingLightGroup : pulse
lsref CeilingLightGroup : csa
lsref CeilingLightGroup : pb
lsref DimmableCeilingLightGroup : dimmer

Problem Description

Part 3: Informal Needs

Need *PD_20*

Information : This part contains the needs for a new light control system for the fourth floor of Building 32 of the University of Kaiserslautern.

Need *PD_21*

Information : In Section 3.1, functional needs are listed and in Section 3.2 non-functional needs are listed.

3.1 Functional Needs

Need *PD_22*

Information : The functional needs are split into two groups, user needs and facility manager needs, depending on the person who has expressed them.

3.1.1 User Needs

Need *PD_23*

Information : The user needs are labelled by U<number>.

Need *PD_24*

Information : At first, general user needs are listed, which are demanded for each kind of room:

Need *U1*

Information : If a person occupies a room, there has to be safe illumination, if nothing else is desired by the chosen light scene.

Postreferences : lsref Area : T_A1
lsref Area : T_A2
lsref Area : safeLight
lsref Area : noSaveLight
rsref Area : R_A3
msref Room : M_R3
lsref Area : evalSaveLight
lsref Room : occRoom
msref Room : M_R1
msref Area : M_A5

Need *U2*

Information : As long as the room is occupied, the chosen light scene has to be maintained.

Postreferences : lsref Room : curLightScene
msref Room : M_R7
lsref Room : occRoom
lsref Room : evalLightScene

Need *U3*

Information	: If the room is reoccupied within T1 minutes after the last person has left the room, the chosen light scene has to be reestablished.
Postreferences	: lsref NoMalfunctionOccupancy : delay lsref NoMalfunctionOccupancy : areaEmpty lsref NoMalfunctionOccupancy : occupied msref NoMalfunctionOccupancy : M_NMO1 lsref Room : occRoom msref Room : M_R5

Need *U4*

Information	: If the room is reoccupied after more than T1 minutes since the last person has left the room, the default light scene has to be established.
Postreferences	: lsref Room : curDefaultLightScene lsref NoMalfunctionOccupancy : delay lsref NoMalfunctionOccupancy : areaEmpty lsref NoMalfunctionOccupancy : occupied msref NoMalfunctionOccupancy : M_NMO1 lsref Room : occRoom msref Room : M_R6

Need *U5*

Information	: For each room, the chosen light scene can be set by using the room control panel.
Postreferences	: lsref RoomCtrlPanel : chosenLightScene msref RoomCtrlPanel : M_RCP1 lsref Room : CPRoom

Need *U6*

Information	: For each room, the default light scene can be set by using the room control panel.
Postreferences	: lsref RoomCtrlPanel : defaultLightScene msref RoomCtrlPanel : M_RCP1 lsref Room : CPRoom msref Room : M_R2

Need *U7*

Information	: For each room, the value T1 can be set by using the room control panel.
Postreferences	: lsref RoomCtrlPanel : reoccDelay msref RoomCtrlPanel : M_RCP1 lsref NoMalfunctionOccupancy : delay lsref NoMalfunctionOccupancy : areaEmpty lsref NoMalfunctionOccupancy : occupied lsref Room : CPRoom msref Room : M_R4

Need *U8*

Information : If any outdoor light sensor or the motion detector of a room does not work correctly, the user of this room has to be informed.

Postreferences : lsref InformPerson : indicator
lsref InformPerson : T_MinInf
lsref InformPerson : T_MaxInf
lsref InformPerson : T_InfDelayNew
lsref InformPerson : informCondEnv
lsref InformPerson : informCondMach
lsref InformPerson : personInformed
lsref RoomCtrlPanel : indicatorMalFct
lsref Area : sensorActMalFct
lsref Room : OLSMalFct

Need *U9*

Information : The room control panel for an office should contain at least:

(i) a possibility to set each ceiling light group

(ii) a possibility to set the chosen and the default light scene

(iii) a possibility to set T1

Postreferences : lsref RoomCtrlPanel : wallCeilLightSwitch
lsref RoomCtrlPanel : winCeilLightSwitch
lsref RoomCtrlPanel : winCeilLightDimmer
lsref RoomCtrlPanel : wallCeilLightDimmer
lsref RoomCtrlPanel : reoccDelay
msref RoomCtrlPanel : M_RCP1
lsref NoMalfunctionOccupancy : delay
lsref NoMalfunctionOccupancy : areaEmpty
lsref NoMalfunctionOccupancy : occupied
lsref Room : CPRoom
msref Room : M_R4

Need *PD_25*

Information : The user needs concerning the offices are:

Need *U10*

Information : The ceiling light groups should be maintained by the control system depending on the current light scene.

Postreferences : msref Room : M_R7

Need *U11*

Information : A room control panel in an office should be movable as is a telephone.

Postreferences : rsref Office : R_O1

Need *PD_26*

Information : The user needs for the remaining rooms are:

Need *U12*

Information : In all other rooms, the room control panel should be installed near a door leading to the hallway section.
Postreferences : rsref NoOffice : R_NO1

Need *PD_27*

Information : The user needs for the hallway sections are:

Need *U13*

Information : When a hallway section is occupied by a person, there has to be safe illumination.
Postreferences : lsref Area : T_A1
lsref Area : T_A2
lsref Area : safeLight
lsref Area : evalSaveLight
rsref Area : R_A3

Need *U14*

Information : Before a person enters one hallway section from another one or from a staircase, the hallway section ceiling light group in the section being entered has to be on.
Postreferences : lsref HallwaySection : T_HS1
lsref HallwaySection : nearHallwayDoor

3.1.2 Facility Manager Needs

Need *PD_28*

Information : The facility manager needs are numbered by FM<number>.

Need *FM1*

Information : Use daylight to achieve the desired light setting of each room and each hallway section whenever possible.
Postreferences : lsref Area : dayLight
lsref Area : evalSaveLight
lsref Room : evalLightScene

Need *FM2*

Information : The ceiling light group in each hallway section has to be off when the hallway section has been unoccupied for at least T2 minutes.

Postreferences : lsref AreaCtrlPanelFM : unoccDelay
lsref NoMalfunctionOccupancy : delay
lsref NoMalfunctionOccupancy : areaEmpty
lsref NoMalfunctionOccupancy : occupied
msref NoMalfunctionOccupancy : M_NMO1
lsref Area : occFM
lsref Area : T_A1
msref Area : M_A6

Need *FM3*

Information : The ceiling light groups in a room have to be off when the room is unoccupied for at least T3 minutes.

Postreferences : lsref AreaCtrlPanelFM : unoccDelay
lsref NoMalfunctionOccupancy : delay
lsref NoMalfunctionOccupancy : areaEmpty
lsref NoMalfunctionOccupancy : occupied
msref NoMalfunctionOccupancy : M_NMO1
lsref Area : occFM
lsref Area : T_A1
msref Area : M_A6

Need *FM4*

Information : For each hallway section, the value T2 can be set by using the facility manager control panel.

Postreferences : lsref AreaCtrlPanelFM : unoccDelay
msref AreaCtrlPanelFM : M_ACPFM1
lsref NoMalfunctionOccupancy : delay
lsref NoMalfunctionOccupancy : areaEmpty
lsref NoMalfunctionOccupancy : occupied
lsref Area : occFM
lsref Area : CPFM

Need *FM5*

Information : For each room, the value T3 can be set by using the facility manager control panel.

Postreferences : lsref AreaCtrlPanelFM : unoccDelay
msref AreaCtrlPanelFM : M_ACPFM1
lsref NoMalfunctionOccupancy : delay
lsref NoMalfunctionOccupancy : areaEmpty
lsref NoMalfunctionOccupancy : occupied
lsref Area : occFM
lsref Area : CPFM

Need *FM6*

Information	: The facility manager can turn off the ceiling light groups in a room or hallway section that is not occupied.
Postreferences	: lsref AreaCtrlPanelFM : lightOff msref AreaCtrlPanelFM : M_ACPFM1 lsref Area : occFM lsref Area : CPFM lsref Area : T_A1

Need *FM7*

Information	: If a malfunction occurs, the facility manager has to be informed.
Postreferences	: lsref InformPerson : T_MinInf lsref InformPerson : T_MaxInf lsref InformPerson : T_InfDelayNew lsref InformPerson : indicator lsref InformPerson : informCondEnv lsref InformPerson : informCondMach lsref InformPerson : personInformed lsref AreaCtrlPanelFM : indicatorMalFct

Need *FM8*

Information	: If a malfunction occurs, the control system supports the facility manager in finding the reason.
Postreferences	: rsref Floor : R_F2

Need *FM9*

Information	: The system provides reports on current and past energy consumption.
Postreferences	: rsref Floor : R_F3

Need *FM10*

Information	: All malfunctions and unusual conditions are stored and reported on request.
Postreferences	: rsref Floor : R_F4

Need *FM11*

Information	: Malfunctions that the system cannot detect can be entered manually.
Postreferences	: lsref MalfunctionOccupancy : manualEnteredOcc
	lsref MalfunctionOccupancy : stateManualEnteredOcc
	lsref MalfunctionSensor : manualEnteredMal
	lsref MalfunctionSensor : stateManualEnteredMal
	lsref MalfunctionSensor : automaticDetMal
	lsref MalfunctionActuator : manualEnteredMal
	lsref MalfunctionSensor : malDetEnv
	lsref MalfunctionSensor : malDetMach
	lsref MalfunctionActuator : malDetEnv
	lsref MalfunctionActuator : malDetMach
	lsref MalfunctionActuator : stateManualEnteredMal
	lsref MalfunctionActuator : automaticDetMal
	msref MalfunctionActuator : M_MA1
	msref MalfunctionSensor : M_MS3
	lsref MalfunctionOccupancy : manualEnteredMal
	lsref MalfunctionOccupancy : stateManualEnteredMal
	lsref MalfunctionOccupancy : automaticDetMal
	msref MalfunctionOccupancy : M_MO1

3.2 Non-Functional Needs

Need *PD_29*

Information	: The non-functional needs are split into several groups according to the aspect they are dealing with. They are labelled by NF<number>.

3.2.1 Fault Tolerance

Need *PD_30*

Information : In any case of failure the system shall provide a stepwise degradation of functionality down to manual operability.

Postreferences : lsref MalfunctionSensor : manualEnteredVal
lsref MalfunctionSensor : automaticDetermValue
lsref MalfunctionSensor : valueMal
lsref MalfunctionSensor : malDetEnv
lsref MalfunctionSensor : malDetMach
lsref MalfunctionSensor : stateManualEnteredValue
msref MalfunctionSensor : M_MS4
lsref MalfunctionActuator : malDetEnv
lsref MalfunctionActuator : malDetMach
lsref EnteredValue : nearestValue
msref EnteredValue : M_EV1
lsref CeilingLightGroup : manualControl
lsref MalfunctionOccupancy : stateManualEnteredOcc
lsref MalfunctionOccupancy : automaticDetermValue
lsref MalfunctionOccupancy : occMal
msref MalfunctionOccupancy : M_MO2
lsref Occupancy : occUsed
msref Occupancy : M_O1
lsref Area : manualControl
msref MalfunctionOccupancy : M_MO1

Need *PD_31*

Information : Needs in the case of a malfunction of the outdoor light sensor:

Need *NF1*

Information : If any outdoor light sensor does not work correctly, the control system for rooms should behave as if the outdoor light sensor had been submitting the last correct measurement of the outdoor light constantly.

Postreferences : lsref MalfunctionSensor : automaticDetermValue
lsref OutdoorLightSensor : lastCorrectOutLight
rsref OutdoorLightSensor : R_OLS1
lsref Floor : outLightSensorFaultMach

Need *NF2*

Information : If any outdoor light sensor does not work correctly, the default light scene for all rooms is that all ceiling light groups are on.
Postreferences : lsref MalfunctionSensor : automaticDetermValue
lsref Room : curDefaultLightScene
lsref Room : malDefaultLightScene
lsref Floor : outLightSensorFaultMach
msref Floor : M_F2
lsref Room : OLSMalFct

Need *NF3*

Information : If any outdoor light sensor does not work correctly and a hallway section is occupied, the ceiling light group in this hallway section has to be on
Postreferences : lsref MalfunctionSensor : automaticDetermValue
lsref Floor : TmalReact
lsref Floor : outLightSensorFaultMach

Need *PD_32*

Information : Needs in the case of a malfunction of the motion detector:

Need *NF4*

Information : If any motion detector of a room or a hallway section does not work correctly, the control system should behave as if the room or the hallway section were occupied.
Postreferences : lsref MalfunctionSensor : automaticDetermValue
lsref MalfunctionOccupancy : malOccDetMach
lsref MalfunctionOccupancy : malOccDetEnv
lsref MalfunctionOccupancy : automaticDetermValue
lsref MalfunctionOccupancy : occMal
msref Area : M_A4

Need *PD_33*

Information : Needs in a worst-case failure of the control system:

Need *NF5*

Information : If the ceiling light group in a hallway section is controllable neither automatically nor manually, the ceiling light group of this hallway section has to be on.
Postreferences : lsref HallwaySection : T_HS2
lsref HallwaySection : hwLightUncontroll

3.2.2 Safety and Legal Aspects

Need *NF6*

Information : All hardware connections are made according to DIN standards.
Postreferences : rsref Floor : R_F5

Need *NF7*

Information	: No hazardous conditions for persons, inventory, or building are allowed.
Postreferences	: rsref Floor : R_F6

3.2.3 User Interface

Need *NF8*

Information	: The control panels are easy and intuitive to use.
Postreferences	: rsref RoomCtrlPanel : R_RCP1

Need *NF9*

Information	: The system issues warnings on unreasonable inputs.
Postreferences	: lsref EnteredValue : checkTime
	lsref EnteredValue : entered
	lsref EnteredValue : checked
	lsref EnteredValue : unreasonableInput
	lsref EnteredValue : nearestValue
	lsref InformPerson : indicator
	lsref InformPerson : T_MinInf
	lsref InformPerson : T_MaxInf
	lsref InformPerson : T_InfDelayNew
	lsref InformPerson : informCondEnv
	lsref InformPerson : informCondMach
	lsref InformPerson : personInformed
	lsref AreaCtrlPanelFM : indicatorUnreasonInp
	lsref RoomCtrlPanel : indicatorUnreasonInp

Global Signature

Domain $LIGHT\_VALUES = \{1 \ldots 10000\} \subseteq NAT$

NL	: light values
Intention	: This domain contains the natural numbers between 1 and 10000. They represent the possible light intensity values in lux that are considered in this problem specification.
Prereferences	: akref Table1
Postreferences	: lsref LightScene : ambLightLevel lsref OutdoorLightSensor : OUT_LIGHT_VALUES lsref Area : dayLight lsref Area : evalSaveLight lsref Room : evalLightScene

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

Intention	: This domain contains the natural numbers 0 and 1. They represent the binary values.
Prereferences	: akref Table1 akref Table2
Postreferences	: lsref BinarySensor : Sensor dkref BinarySensor : D_BS2 dkref BinarySensor : D_BS3 lsref ToggleActuator : Actuator lsref BinaryActuator : RangeActuator dkref BinaryActuator : D_BA1 lsref AreaCtrlPanelFM : lightOff lsref RoomCtrlPanel : winCeilLightSwitch lsref RoomCtrlPanel : wallCeilLightSwitch lsref RoomCtrlPanel : lightSceneSwitch lsref MalfunctionOccupancy : manualEnteredOcc lsref MalfunctionSensor : manualEnteredMal lsref MalfunctionActuator : manualEnteredMal lsref Area : evalSaveLight lsref MalfunctionOccupancy : manualEnteredMal

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

Intention	: This domain contains the natural numbers between 0 to 100. They represent the possible percentual values considered in this problem specification.
Prereferences	: akref Table2
Postreferences	: lsref Dimmer : RangeActuator lsref RoomCtrlPanel : wallCeilLightDimmer lsref RoomCtrlPanel : winCeilLightDimmer lsref Room : evalLightScene msref Room : M_R10

Group Datatypes

Intention

This group contains all description classes that are used as datatypes.

This are the following description classes:

- *LightScene*

Dies muß noch üeberarbeite werden, d.h. es muß genauer überlegt werden, welche Art von Klassen in dieser Gruppe enthalten sind.

Template Declaration LightScene

Intention

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

- the name of a light scene.
  This is represented by the timed function *name*.
- the desired ambient light level in a room.
  This is represented by the timed function *ambLightLevel*.
- the option of a light scene indicating in which way the desired ambient light level should be achieved.
  This is represented by the timed function *option*.

Postreferences : lsref RoomCtrlPanel : defaultLightScene
msref RoomCtrlPanel : I_M_RCP10
lsref Room : curLightScene
lsref Room : curDefaultLightScene
lsref Room : malDefaultLightScene
lsref Room : noMalDefaultLightScene
lsref Room : determineLightScene
lsref Room : evalLightScene
msref Room : M_R1
msref Room : M_R7
msref Room : M_R8
msref Room : M_R8b

Signature

Value *WINDOW* : *OPTIONS*

Intention : This value represents the option of a light scene meaning that at first the ceiling light group near the window should be used to achieve the desired ambient light level and then the other ceiling light group.

Postreferences : lsref LightScene : CEIL_LIGHT_GROUPS
msref Room : M_R8

Value *WALL* : *OPTIONS*

Intention : This value represents the option of a light scene meaning that at first the ceiling light group near the wall to the hallway should be used to achieve the desired ambient light level and then the other ceiling light group.

Postreferences : lsref LightScene : CEIL_LIGHT_GROUPS
msref Room : M_R8b

Value *BOTH* : *OPTIONS*

Intention	: This value represents the option of a light scene meaning that both ceiling light groups should be used equally to achieve the desired ambient light level
Postreferences	: msref RoomCtrlPanel : I_M_RCP10 msref Room : M_R1

Domain $OPTIONS = \{WINDOW, WALL, BOTH\}$

Intention	: This domain contains the possible options that can be specified for a light scene.
Postreferences	: lsref LightScene : option lsref LightScene : CEIL_LIGHT_GROUPS

Domain $CEIL\_LIGHT\_GROUPS = \{WINDOW, WALL\} \subseteq OPTIONS$

Intention	: This domain contains the possible ceiling light groups in a room that are considered by a light scene.
Prereferences	: akref BD_8
Postreferences	: lsref Room : evalLightScene

Timed Function $name \rightarrow STRING$

NL	: light scene name
Intention	: This function represents the name of the light scene.
Scope	: mh
Postreferences	: msref RoomCtrlPanel : I_M_RCP8 msref Room : M_R1

Timed Function $ambLightLevel \rightarrow LIGHT\_VALUES$

NL	: ambient light level
Intention	: This function represents the desired ambient light level in a room specified by a light scene.
Scope	: mh
Postreferences	: msref RoomCtrlPanel : I_M_RCP9 msref Room : M_R1

Timed Function $option \rightarrow OPTIONS$

NL	: light scene option
Intention	: This function represents the option specified by a light scene indicating in which way the specified ambient light level should be achieved.
Scope	: mh
Postreferences	: msref RoomCtrlPanel : I_M_RCP10 msref Room : M_R1

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.

This are the following description classes:

- *EnteredValue*

Template Declaration 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 a valid value and returns the valid value which is *nearest* to e. Whether an entered value is a valid one is represented by the timed predicate *unreasonableInput*

It is assumed that 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 machine. Thus, this device is part of the machine. In our case this device is the control panel, which still has to be developed. Therefore the timed function *entered* has the scope *ev* and the timed function *checked* has the scope *mh*. Furthermore the optative statements are machine specifications.

Postreferences : lsref MalfunctionSensor : manualEnteredVal
lsref MalfunctionSensor : manualEnteredMal
lsref MalfunctionActuator : manualEnteredMal
lsref AreaCtrlPanelFM : lightOff
lsref AreaCtrlPanelFM : unoccDelay
lsref RoomCtrlPanel : chosenLightScene
lsref RoomCtrlPanel : winCeilLightSwitch
lsref RoomCtrlPanel : wallCeilLightSwitch
lsref RoomCtrlPanel : winCeilLightDimmer
lsref RoomCtrlPanel : wallCeilLightDimmer
lsref RoomCtrlPanel : lightSceneSwitch
lsref RoomCtrlPanel : reoccDelay
lsref RoomCtrlPanel : defaultLightScene
lsref MalfunctionOccupancy : manualEnteredOcc
lsref MalfunctionOccupancy : manualEnteredMal

Formal Parameters

Sort *VALID_INPUTS*

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

Postreferences : lsref EnteredValue : checked
lsref EnteredValue : nearestValue

Signature

Time Constant *checkTime*

NL	: time for checking
Intention	: This time constant represents the maximal time needed to check a value, entered by a person, using the function *nearestValue*.
Scope	: ev
Prereferences	: pdref NF9
Postreferences	: msref EnteredValue : M_EV1 dkref MalfunctionSensor : D_MS1 dkref MalfunctionSensor : D_MS2

Timed Function *entered* $\rightarrow$ *STRING*

NL	: entered value
Intention	: This function represents the value entered by a person. The value is represented as a string.
Scope	: ev
Prereferences	: pdref NF9
Postreferences	: msref EnteredValue : M_EV1 dkref MalfunctionSensor : D_MS1 dkref MalfunctionSensor : D_MS2

Timed Function *checked* $\rightarrow$ *VALID_INPUTS*

NL	: checked value
Intention	: This function represents the value decided by the machine to use. *checked* is determined out of *entered* using the function *nearestValue*.
Scope	: mh
Prereferences	: pdref NF9
Postreferences	: msref MalfunctionSensor : M_MS4
	msref AreaCtrlPanelFM : I_M_ACPFM1
	msref AreaCtrlPanelFM : I_M_ACPFM2
	msref MalfunctionOccupancy : M_MO2
	msref Room : M_R3
	msref Room : M_R7
	msref EnteredValue : M_EV1
	msref MalfunctionSensor : M_MS3
	msref Room : M_R2
	msref Room : M_R5
	msref Room : M_R6
	msref RoomCtrlPanel : I_M_RCP7
	msref MalfunctionActuator : M_MA1
	msref RoomCtrlPanel : I_M_RCP1
	msref RoomCtrlPanel : I_M_RCP2
	msref RoomCtrlPanel : I_M_RCP3
	msref RoomCtrlPanel : I_M_RCP4
	msref RoomCtrlPanel : I_M_RCP5
	msref RoomCtrlPanel : I_M_RCP6
	msref RoomCtrlPanel : I_M_RCP8
	msref RoomCtrlPanel : I_M_RCP9
	msref RoomCtrlPanel : I_M_RCP10
	msref MalfunctionOccupancy : I_M_MO1
	msref Room : M_R4
	msref MalfunctionOccupancy : M_MO1
	msref MalfunctionOccupancy : I_M_MO5
	rsref Area : R_A1
	msref Area : M_A3
	msref Room : M_R8
	msref Room : M_R8b
	msref Room : M_R9

Timed Predicate *unreasonableInput*

Intention	: This predicate is true if and only if the last entered value does not belong to the sort VALID_INPUTS, i.e., if it is an *unreasonable input*.
Scope	: mh
Prereferences	: pdref NF9
Postreferences	: msref EnteredValue : I_M_EV1 msref AreaCtrlPanelFM : M_ACPFM1 msref RoomCtrlPanel : M_RCP1

Function $nearestValue : STRING \rightarrow VALID\_INPUTS$

Intention	: This function returns the value which is the *nearest* one in the domain VALID_INPUTS w.r.t. the first argument.
Scope	: mh
Prereferences	: pdref NF9 pdref PD_30
Postreferences	: msref EnteredValue : M_EV1 msref MalfunctionSensor : M_MS1 msref MalfunctionSensor : M_MS2

Machine Specifications

Property *I_M_EV1*

Formal	: $\neg unreasonableInput$
NL	: Initially there is no unreasonable input, i.e., initially the last entered value belongs to the sort VALID_INPUTS.

Property *M_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*.
Prereferences	: pdref PD_30

Group Sensors

Intention

This group contains all description classes that contain definitions related to sensors. This are the following description classes:

- *NoMalfunctionSensor*
- *MalfunctionSensor*
- *Sensor*
- *OutdoorLightSensor*
- *BinarySensor*
- *StatusLine*
- *DoorClosedContact*
- *MotionDetector*

Template Declaration NoMalfunctionSensor

Intention

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

It is distinguished between three values:

- the value of the considered phenomenon existing in the real world, represented by the timed function *envEntity*, e.g. the temperature in a room,
- the measured analog value of this real world value, represented by the timed function *measuredEntity*, and
- the digital value which is a conversion of the measured analog value, represented by the 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 phenomenon in the real world is represented by a function having the scope *environment controlled, hidden for the machine* and the value measured by a sensor, namely *convertedEntity*, has the scope *environment controlled, visible for the machine*. Secondly, the properties D_NMS_1 and D_NMS_2 describing the behavior of a sensor are characterized as domain knowledge. Otherwise, if a sensor does not exist, they have to be stated in the optative mood.

Postreferences : lsref Sensor : noMalSens

Formal Parameters

Sort *ENV_DOMAIN*

Intention	: This domain contains the possible values of the measured phenomenon of the real world.
Postreferences	: lsref NoMalfunctionSensor : envEntity lsref NoMalfunctionSensor : modifyReaction

Sort *MEASURED_DOMAIN*

Intention	: This domain contains the possible measured analog values.
Postreferences	: lsref NoMalfunctionSensor : measuredEntity lsref NoMalfunctionSensor : modifyReaction lsref NoMalfunctionSensor : modifyConversion

Sort *CONVERTED_DOMAIN*

Intention	: This domain contains the possible digital values, converted from an analog value.
Postreferences	: lsref NoMalfunctionSensor : convertedEntity lsref NoMalfunctionSensor : modifyConversion

Signature

Time Constant *reactionTime*

Intention	: This time constant represents the *reaction time* of a 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, excluding conversion time.
Scope	: ev
Prereferences	: akref BD_17 akref Table1
Postreferences	: dkref NoMalfunctionSensor : D_NMS1 dkref OutdoorLightSensor : D_OLS2 dkref StatusLine : D_SL1 dkref DoorClosedContact : D_DCC1 dkref MotionDetector : D_MD1

Time Constant *conversionTime*

Intention	: This time constant represents the *conversion time* of a sensor. The *conversion time* is the time needed to convert a measured analog value (*measuredEntity*) into the corresponding digital value (*convertedEntity*).
Scope	: ev
Prereferences	: akref BD_17 akref Table1
Postreferences	: dkref NoMalfunctionSensor : D_NMS2 dkref OutdoorLightSensor : D_OLS1 dkref BinarySensor : D_BS1

Timed Function $envEntity \rightarrow ENV\_DOMAIN$

Intention	: This function represents the value of the measured phenomenon in the real world. Note that this function represents the phenomenon as it is in the real world and not how it is seen by the machine.
Scope	: eh
Prereferences	: akref BD_17
Postreferences	: dkref NoMalfunctionSensor : D_NMS1 dkref CeilingLightGroup : D_CLG1 dkref CeilingLightGroup : D_CLG2 dkref CeilingLightGroup : D_CLG3 rsref Floor : R_F1 rsref Area : R_A1 rsref Area : R_A2 rsref HallwaySection : R_HS1 rsref HallwaySection : R_HS2

Timed Function $measuredEntity \rightarrow MEASURED\_DOMAIN$

Intention	: This function represents the measured analog value of the phenomenon of the real world.
Scope	: eh
Prereferences	: akref BD_17
Postreferences	: dkref NoMalfunctionSensor : D_NMS1 dkref NoMalfunctionSensor : D_NMS2

Timed Function $convertedEntity \rightarrow CONVERTED\_DOMAIN$

Intention	: This function represents the digital value derived from the measured analog value.
Scope	: ev
Prereferences	: akref BD_17
Postreferences	: dkref NoMalfunctionSensor : D_NMS2 msref Sensor : M_S1 msref CeilingLightGroup : M_CLG1

Function $modifyReaction : ENV\_DOMAIN \rightarrow MEASURED\_DOMAIN$

Intention	: This function represents the way in which a value of the phenomenon of the real world (*envEntity*) is modified during the reaction of a sensor resulting in a measured analog value *measuredEntity*.
Scope	: eh
Prereferences	: akref BD_17
Postreferences	: dkref NoMalfunctionSensor : D_NMS1 dkref BinarySensor : D_BS2

Function $modifyConversion : MEASURED\_DOMAIN \rightarrow CONVERTED\_DOMAIN$

Intention	: This function represents the way in which a measured analog value *measuredEntity* is modified during the conversion into a digital value (*convertedEntity*).
Scope	: eh
Prereferences	: akref BD_17
Postreferences	: dkref NoMalfunctionSensor : D_NMS2 dkref BinarySensor : D_BS3

Domain Knowledge

Property *D_NMS1*

Formal	: $\Box(envEntity \rhd^{modifyReaction}_{reactionTime}\ measuredEntity)$
NL	: The measured analog value is always derived from the value of the phenomenon of the real world using the function *modifyReaction*. A change of the value in the real world is always propagated within the time *reactionTime*.
Prereferences	: akref BD_17

Property *D_NMS2*

Formal	: $\Box(measuredEntity \rhd^{modifyConversion}_{conversionTime} convertedEntity)$
NL	: A digital value is always derived from a measured analog value using the function *modifyConversion*. A change of the measured analog value is always propagated within the time *conversionTime*.
Prereferences	: akref BD_17

Template Declaration MalfunctionSensor

Intention

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

The following topics are considered:

- A malfunction of a sensor can be detected automatically by the machine or it can be entered manually. To specify this topic the following signature entities *manualEnteredMal*, *T_MalDetDelay*, *malDetEnv*, *malDetMach*, *stateManualEnteredMal*, *automaticDetMal* and the following properties I_A_MS1, I_A_MS3, A_MS1 - A_MS_4, I_M_MS1, M_MS1 - M_MS3 are used.

- 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. To specify this topic the following signature entities *manualEnteredVal*, *automaticDetermValue*, *valueMal*, *stateManualEnteredValue*, I_A_MS2, and the following properties M_MS8 are used.

  This is our interpretation of *stepwise degradation* in the case that a sensor has a malfunction.

Postreferences : lsref Sensor : malSens

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 a sensor.
Postreferences	: lsref MalfunctionSensor : manualEnteredVal lsref MalfunctionSensor : automaticDetermValue lsref MalfunctionSensor : valueMal

Signature

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

Intention	: This object represents the value that should be used in the case of a malfunction of a sensor. This value is entered manually by a user. The entered value has to be an element of the domain MAL_CONVERTED_DOMAIN.
Prereferences	: pdref PD_30
Postreferences	: msref MalfunctionSensor : M_MS4

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

Intention : This object represents a value entered manually by a person indicating whether a sensor has a malfunction or not.

- 1 represents that a sensor has a malfunction
- 0 represents that a sensor has no malfunction

Note that the value of this object is only meaningful if the timed predicate *stateManualEnteredMal* is true.

Prereferences : pdref FM11

Postreferences : msref MalfunctionSensor : M_MS3
dkref MalfunctionSensor : D_MS1
dkref MalfunctionSensor : D_MS2
msref MalfunctionSensor : M_MS1
msref MalfunctionSensor : M_MS2

Time Constant *T_MalDetDelay*

Intention : This time constant represents the maximal delay between the time point a sensor has a malfunction and the time point the machine assumes a malfunction of this sensor.

Scope : ev

Postreferences : dkref MalfunctionSensor : D_MS1
dkref MalfunctionSensor : D_MS2
dkref MalfunctionSensor : D_MS3
dkref MalfunctionSensor : D_MS4
msref MalfunctionSensor : I_M_MS1

Timed Function *automaticDetermValue* $\rightarrow$ *MAL_CONVERTED_DOMAIN*

Intention : This function represents the value of a sensor determined automatically by the machine in the case of a malfunction of a sensor.

Scope : mh

Prereferences : pdref PD_30
pdref NF1
pdref NF2
pdref NF3
pdref NF4

Postreferences : msref MalfunctionSensor : M_MS4
rsref OutdoorLightSensor : R_OLS1

Timed Function *valueMal* $\rightarrow$ *MAL_CONVERTED_DOMAIN*

Intention	: This function represents the value of a sensor used by the machine in the case of a malfunction of a sensor. It is computed based on the automatically determined and the manually entered value.
Scope	: mh
Prereferences	: pdref PD_30
Postreferences	: msref MalfunctionSensor : M_MS4 msref Sensor : M_S2

Timed Predicate *malDetEnv*

Intention	: This predicate is true if and only if a 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
Prereferences	: pdref PD_30 pdref FM11
Postreferences	: dkref MalfunctionSensor : D_MS1 dkref MalfunctionSensor : D_MS2 dkref MalfunctionSensor : I_D_MS3 dkref MalfunctionSensor : D_MS3 dkref MalfunctionSensor : D_MS4

Timed Predicate *malDetMach*

Intention	: This predicate is true if and only if a machine assumes that a sensor existing in the real world has a malfunction. Note that this predicate represents the value as it seen by the machine.
Scope	: mh
Prereferences	: pdref PD_30 pdref FM11
Postreferences	: msref Sensor : M_S1 msref Sensor : M_S2 msref Floor : M_F1 msref MalfunctionSensor : M_MS3 msref MalfunctionSensor : I_M_MS1 msref Floor : M_F13 msref OfficeNeighbor : M_ON1 msref NoOfficeNeighbor : M_NON1 msref Floor : M_F3 msref Floor : M_F4 msref Floor : M_F14 msref Floor : M_F15 msref Floor : M_F16 msref Floor : M_F17 msref Floor : M_F18 msref Floor : M_F19 msref Floor : M_F20 msref Floor : M_F21 msref Floor : M_F22 msref Floor : M_F23 msref Floor : M_F24 msref Floor : M_F25 msref Floor : M_F6 msref Floor : M_F7 msref Floor : M_F8 msref Floor : M_F9 msref Floor : M_F10 msref Floor : M_F11 msref Floor : M_F12 msref HallwaySection : M_HS4 msref Room : M_R13 msref Room : M_R14

Timed Predicate *stateManualEnteredValue*

Intention	: This predicate is true if and only if a value, that should be used in the case of a malfunction of a sensor, has been entered manually by a user.
Scope	: ev
Prereferences	: pdref PD_30
Postreferences	: msref MalfunctionSensor : M_MS4 dkref MalfunctionSensor : I_D_MS2

Timed Predicate *stateManualEnteredMal*

Intention	: This predicate is true if and only if it has been manually entered whether there is a malfunction of a sensor or whether there is no malfunction of it. Note the difference between this predicate and the object *manualEnteredMal*.
Scope	: ev
Prereferences	: pdref FM11
Postreferences	: dkref MalfunctionSensor : I_D_MS1 msref MalfunctionSensor : M_MS3 dkref MalfunctionSensor : D_MS1 dkref MalfunctionSensor : D_MS2

Timed Predicate *automaticDetMal*

Intention	: This predicate is true if and only if a malfunction of a sensor has been detected automatically, i.e. not by entering it manually.
Scope	: ev
Prereferences	: pdref FM11
Postreferences	: msref MalfunctionSensor : M_MS3 dkref MalfunctionSensor : D_MS1 dkref MalfunctionSensor : D_MS2

Domain Knowledge

Property *I_D_MS1*

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

Property *I_D_MS2*

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

Property *I_D_MS3*

Formal	: $\neg malDetEnv$
NL	: Initially a sensor has no malfunction.

Property *D_MS1*

Formal : $\Box(malDetEnv \Rightarrow_{\leq T\_MalDetDelay}$
$(\neg stateManualEnteredMal \wedge automaticDetMal) \vee$
$(stateManualEnteredMal \wedge$
$\blacksquare_{\leq manualEnteredMal.checkTime}(manualEnteredMal.entered = \text{“1”})))$

NL : Whenever a sensor existing in the real world has constantly a malfunction for at least *T_MalDetDelay* milliseconds then within this time the conditions are fulfilled that the machine assumes a malfunction of this sensor, i.e.,

- either the machine has detected a malfunction automatically and it is not entered manually whether there is a malfunction or not
- or it is entered manually whether there is a malfunction or not and in the last *manualEnteredMal.checkTime* milliseconds it has been entered that there is a malfunction, i.e. *manualEnteredMal.entered* = ”1”.

Property *D_MS2*

Formal : $\Box(\neg malDetEnv \Rightarrow_{\leq T\_MalDetDelay}$
$(\neg stateManualEnteredMal \wedge \neg automaticDetMal) \vee$
$(stateManualEnteredMal \wedge$
$\blacksquare_{\leq manualEnteredMal.checkTime}(manualEnteredMal.entered \neq \text{“1”})))$

NL : Whenever a sensor existing in the real world has constantly no malfunction for at least *T_MalDetDelay* milliseconds then within this time the conditions are fulfilled that the machine assumes no malfunction of this sensor, i.e.,

- either the machine has not detected a malfunction automatically and it is not entered manually whether there is a malfunction or not
- or it is entered manually whether there is a malfunction or not and in the last *manualEnteredMal.checkTime* milliseconds it has not been entered that there is a malfunction, i.e. *manualEnteredMal.entered* ≠ ”1”.

Property *D_MS3*

Formal : $\Box(\underset{\leq T\_MalDetDelay}{\nabla} malDetEnv)$

NL : Each time *malDetEnv* becomes valid, it will be constantly valid for at least *T_MalDetDelay* milliseconds.

Property *D_MS4*

Formal : $\Box(\nabla_{\leq T\_MalDetDelay} \neg malDetEnv)$

NL : Each time *malDetEnv* becomes invalid, it will be constantly invalid for at least *T_MalDetDelay* milliseconds.

Machine Specifications

Property *I_M_MS1*

Formal : $\Box_{\leq T\_MalDetDelay} \neg malDetMach$

NL : The first *T_MalDetDelay* milliseconds the machine assumes that a sensor has no malfunction.

Property *M_MS1*

Formal : $\Box(manualEnteredMal.nearestValue(\text{“1”}) = 1)$

NL : A manually entered “1” (as a string) concerning whether a sensor has a malfunction or not is always converted to a 1 as a natural number, i.e. as an element of the domain *BINARY*.

Property *M_MS2*

Formal : $\Box(\forall s \in STRING :$
$(s \neq \text{“1”} \rightarrow manualEnteredMal.nearestValue(s) = 0))$

NL : Each manually entered string concerning whether a sensor has a malfunction or not that differs from “1” (as a string) is always converted to a 0 as a natural number, i.e. as an element of the domain *BINARY*.

Property *M_MS3*

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

NL : The machine assumes a malfunction of a sensor if and only if 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.

Prereferences : pdref FM11

Property *M_MS4*

Formal	: $\Box(stateManualEnteredValue \rightarrow$ $(valueMal = manualEnteredVal.checked) \wedge$ $(\neg stateManualEnteredValue \rightarrow$ $(valueMal = automaticDetermValue)))$
NL	: If a value for the status of a 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 a sensor. Otherwise, the value used by the machine in the case of a malfunction of a sensor is the value determined automatically in the case of a malfunction of a sensor.
Prereferences	: pdref PD_30

Template Declaration Sensor

Intention

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

Postreferences : lsref OutdoorLightSensor : Sensor
lsref BinarySensor : Sensor

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.

Postreferences : lsref Sensor : noMalSens

Sort *MEASURED_DOMAIN*

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

Postreferences : lsref Sensor : noMalSens

Sort *CONVERTED_DOMAIN*

Intention : This domain contains the possible digitally converted measured values in the case that a sensor has no malfunction. This domain corresponds with the domain containing the values the machine can use as sensor values in the case of a malfunction of a sensor.

Postreferences : lsref Sensor : noMalSens
lsref Sensor : malSens
lsref Sensor : usedValue

Signature

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

Intention	: This object represents the part of a sensor describing the behavior of a sensor if it has no malfunction, i.e., if the machine assumes that it has no malfunction.
Postreferences	: dkref OutdoorLightSensor : D_OLS1 dkref OutdoorLightSensor : D_OLS2 dkref BinarySensor : D_BS1 dkref BinarySensor : D_BS2 dkref BinarySensor : D_BS3 dkref StatusLine : D_SL1 dkref DoorClosedContact : D_DCC1 dkref MotionDetector : D_MD1 msref Sensor : M_S1 dkref CeilingLightGroup : D_CLG1 dkref CeilingLightGroup : D_CLG2 dkref CeilingLightGroup : D_CLG3 msref CeilingLightGroup : M_CLG1 rsref Floor : R_F1 rsref Area : R_A1 rsref Area : R_A2 rsref HallwaySection : R_HS1 rsref HallwaySection : R_HS2

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

Intention : This object represents the part of a sensor describing the behavior of a 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 a sensor (*MAL_CONVERTED_DOMAIN*).

Postreferences : msref Sensor : M_S1
msref Sensor : M_S2
msref Floor : M_F1
rsref OutdoorLightSensor : R_OLS1
msref Floor : M_F13
msref OfficeNeighbor : M_ON1
msref NoOfficeNeighbor : M_NON1
msref Floor : M_F3
msref Floor : M_F4
msref Floor : M_F14
msref Floor : M_F15
msref Floor : M_F16
msref Floor : M_F17
msref Floor : M_F18
msref Floor : M_F19
msref Floor : M_F20
msref Floor : M_F21
msref Floor : M_F22
msref Floor : M_F23
msref Floor : M_F24
msref Floor : M_F25
msref Floor : M_F6
msref Floor : M_F7
msref Floor : M_F8
msref Floor : M_F9
msref Floor : M_F10
msref Floor : M_F11
msref Floor : M_F12
msref HallwaySection : M_HS4
msref Room : M_R13
msref Room : M_R14

Timed Function $usedValue \rightarrow CONVERTED\_DOMAIN$

Intention	: This function represents the current value used by the machine as the value of a sensor. This depends on whether there is a malfunction of a sensor or not.
Scope	: ev
Postreferences	: msref Sensor : M_S1 msref Sensor : M_S2 msref HallwaySection : M_HS3 msref Room : M_R11 msref Room : M_R12 dkref HallwaySection : D_HS1 msref Area : M_A5 msref Area : M_A6 msref Room : M_R8 msref Room : M_R8b msref Room : M_R9

Machine Specifications

Property *M_S1*

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

Property *M_S2*

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

Template Declaration OutdoorLightSensor

Intention

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

- the domains of a sensor
- the reaction and conversion time of a sensor
- the automatically determined value in the case of a malfunction

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

Postreferences : lsref Floor : ols1
lsref Floor : ols2
lsref Floor : ols3
lsref Floor : ols4
lsref Floor : ols5
lsref Floor : ols6

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 measured analog values, and using the values of the domain OUT_LIGHT_VALUES as the possible digitally converted measured values.

Signature

Domain $OUT\_LIGHT\_VALUES = \{1 \ldots 10000\} \subseteq LIGHT\_VALUES$

NL : outdoor light values
Intention : This domain contains the possible values measured by an outdoor light sensor.
Prereferences : akref Table1
Postreferences : lsref OutdoorLightSensor : Sensor
lsref OutdoorLightSensor : lastCorrectOutLight

Timed Function *lastCorrectOutLight* $\rightarrow$ *OUT_LIGHT_VALUES*

NL	: last correctly measured outdoor light
Intention	: This function represents the last correctly measured value of the outdoor light intensity.
Scope	: eh
Prereferences	: pdref NF1
Postreferences	: rsref OutdoorLightSensor : R_OLS1

Domain Knowledge

Property *D_OLS1*

Formal	: $\Box(noMalSens.conversionTime = 1000)$
NL	: The conversion time of an outdoor light sensor is 1 second.
Prereferences	: akref Table1

Property *D_OLS2*

Formal	: $\Box(noMalSens.reactionTime = 10)$
NL	: The reaction time of a outdoor light sensor is 10 milliseconds.
Prereferences	: akref Table1

Requirement Specifications

Property *R_OLS1*

Formal	: $\Box(malSens.automaticDetermValue = lastCorrectOutLight)$
NL	: The value of a sensor determined automatically in the case of a malfunction of an outdoor light sensor corresponds with the last correctly measured value of the outdoor light intensity.
Prereferences	: pdref NF1

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

Postreferences : lsref StatusLine : BinarySensor
lsref DoorClosedContact : BinarySensor
lsref MotionDetector : BinarySensor

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 measured analog 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.
Prereferences : akref Table1

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 a sensor, resulting in a measured analog 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 a measured analog value (*measuredEntity*) is modified during the conversion into a digital value (*convertedEntity*) is the identity.

Template Declaration StatusLine

Intention

This class provides all definitions of a sensor determining whether a ceiling light group is on or not. This class is derived from the description class *sensor*.

Note that this is not a sensor measuring the illuminance. Therefore, *on* means that the circuit of the ceiling light group is closed. A *status line* delivers

- a 1 if the circuit of a ceiling light group is closed
- a 0 if the circuit of a ceiling light group is not closed

The following specific parameters of a status line are instantiated :

- the reaction time of a status line

Postreferences : lsref CeilingLightGroup : sl

Base Classes

Class *BinarySensor*

Intention : All definitions of a binary sensor are inherited.

Domain Knowledge

Property *D_SL1*

Formal	: $\Box(noMalSens.reactionTime = 10)$
NL	: The reaction time of a status line is 10 ms.
Prereferences	: akref Table1

Template Declaration DoorClosedContact

Intention

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

A door closed contact delivers a 1 if the door is fully closed, otherwise a 0, i.e., these are the interpretations of the values of the timed function introduced in description class *Sensor*.

Postreferences : lsref Door : doorClosedContact

Base Classes

Class *BinarySensor*

Intention	: All definitions of description class *BinarySensor* are inherited.

Domain Knowledge

Property *D_DCC1*

Formal	: $\Box(noMalSens.reactionTime = 10)$
NL	: The reaction time of a door closed contact is 10 ms.
Prereferences	: akref Table1

Template Declaration MotionDetector

Intention

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

- the reaction time of a status line

A motion detector delivers

- a 1 if a person is moving in the range of the detector
- a 0 otherwise

Postreferences :	lsref HallwaySection : eastMd
	lsref HallwaySection : westMd
	lsref HallwaySection : middleMd
	lsref Room : md
	lsref Staircase : md

Base Classes

Class *BinarySensor*

Intention	: All definitions of description class *BinarySensor* are inherited.

Domain Knowledge

Property *D_MD1*

Formal	: $\Box(noMalSens.reactionTime = 1000)$
NL	: The reaction time of a motion detector is 1 s.
Prereferences	: akref Table1

Group Actuator

Intention

This group contains all description classes that contain definitions related to actuators. This are the following description classes:

- *NoMalfunctionActuator*
- *MalfunctionActuator*
- *Actuator*
- *ToggleActuator*
- *Pulse*
- *RangeActuator*
- *Dimmer*
- *BinaryActuator*
- *ControlSystemActive*
- *IndicatorLight*
- *PushButton*

Template Declaration NoMalfunctionActuator

Intention

This class provides all definitions to describe the behavior of an already existing *actuator* if it has *no malfunction*.

It is distinguished between two values:

- the value of the phenomenon as it exists in the real world represented by the timed function *envEntity*, e.g. the position of a radiator valve
- the value adjusted by the machine represented by the timed function *adjustedEntity*.

Note that this class can be used to describe an actuator that already exists in the environment. This is expressed as follows: The phenomenon in the real world is represented by a function, namely *envEntity*, having the scope *environment controlled, hidden for the machine*, and the value adjusted by an actuator, namely *adjustedEntity*, has the scope *machine controlled, visible for the machine*. If an actuator does not exist, i.e., an actuator itself has to be developed, the scope of *envEntity* has to be *environment controlled, visible for the machine* and the scope of *adjustedEntity* has to be *machine controlled, not visible for the environment*.

Postreferences : lsref Actuator : noMalAct

Formal Parameters

Sort *ENV_DOMAIN*

Intention	: This domain contains the possible values of the phenomenon existing in the real world.
Postreferences	: lsref NoMalfunctionActuator : envEntity

Sort *ADJUSTED_DOMAIN*

Intention	: This domain contains the possible values that can be adjusted by the machine.
Postreferences	: lsref NoMalfunctionActuator : adjustedEntity

Signature

Time Constant *reactionTime*

Intention	: This time constant represents the *reaction time* of an 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 an actuator in the real world (*envEntity*) changes its value *correspondingly*.
Scope	: ev
Prereferences	: akref BD_18 akref Table2
Postreferences	: dkref ToggleActuator : D_TA1 dkref ToggleActuator : D_TA2 dkref Pulse : D_P1 dkref RangeActuator : D_RA1 dkref Dimmer : D_D1 dkref ControlSystemActive : D_CSA1 dkref IndicatorLight : D_IL1 msref CeilingLightGroup : M_CLG1 msref CeilingLightGroup : M_CLG2 msref HallwaySection : M_HS5

Timed Function $envEntity \rightarrow ENV\_DOMAIN$

Intention	: This function represents the value of the phenomenon, i.e. an 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
Prereferences	: akref BD_18
Postreferences	: dkref ToggleActuator : D_TA1 dkref ToggleActuator : D_TA2 dkref RangeActuator : D_RA1 dkref InformPerson : D_IP1 dkref CeilingLightGroup : D_CLG1 dkref CeilingLightGroup : D_CLG3 dkref DimmableCeilingLightGroup : D_DCLG1

Timed Function *adjustedEntity* → *ADJUSTED_DOMAIN*

Intention	: This function represents the value that is adjusted by the machine.
Scope	: mv
Prereferences	: akref BD_18
Postreferences	: dkref ToggleActuator : D_TA1
	dkref ToggleActuator : D_TA2
	dkref RangeActuator : D_RA1
	msref AreaCtrlPanelFM : I_M_ACPFM3
	msref RoomCtrlPanel : I_M_RCP11
	msref InformPerson : M_IP2
	msref InformPerson : M_IP1
	msref InformPerson : M_IP3
	msref AreaCtrlPanelFM : I_M_ACPFM4
	msref RoomCtrlPanel : I_M_RCP12
	msref InformPerson : I_M_IP2
	msref CeilingLightGroup : M_CLG1
	msref CeilingLightGroup : M_CLG2
	msref CeilingLightGroup : M_CLG3
	msref CeilingLightGroup : M_CLG4
	msref Area : M_A5
	msref Area : M_A6
	msref Area : M_A8
	msref Room : M_R8
	msref Room : M_R8b
	msref Room : M_R9
	msref Room : M_R10

Template Declaration MalfunctionActuator

Intention

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

Postreferences : lsref Actuator : malAct

Signature

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

Intention	: This object represents a value entered manually by a user concerning a malfunction of an 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.
Prereferences	: pdref FM11
Postreferences	: msref MalfunctionActuator : M_MA1

Timed Predicate *malDetEnv*

Intention	: This predicate is true if and only if an 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
Prereferences	: pdref PD_30 pdref FM11

Timed Predicate *malDetMach*

Intention	: This predicate is true if and only if the machine assumes that an actuator existing in the real world has a malfunction. Note that this predicate represents the value as it seen by the machine.
Scope	: mh
Prereferences	: pdref PD_30 pdref FM11
Postreferences	: msref MalfunctionActuator : M_MA1 msref HallwaySection : M_HS4 msref Room : M_R13 msref Room : M_R14

Timed Predicate *stateManualEnteredMal*

Intention	: This predicate is true if and only if it has been manually entered whether there is a malfunction of an actuator or whether there is no malfunction of it. Note the difference between this predicate and the object *manualEnteredMal*.
Scope	: mh
Prereferences	: pdref FM11
Postreferences	: msref MalfunctionActuator : I_M_MA1 msref MalfunctionActuator : M_MA1

Timed Predicate *automaticDetMal*

Intention	: This predicate is true if and only if a malfunction of an actuator has been detected automatically, i.e. not by entering it manually.
Scope	: mh
Prereferences	: pdref FM11
Postreferences	: msref MalfunctionActuator : M_MA1

Machine Specifications

Property *I_M_MA1*

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

Property *M_MA1*

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

NL : The machine assumes a malfunction of an actuator if and only if 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

Prereferences : pdref FM11

Template Declaration Actuator

Intention

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

Postreferences : lsref ToggleActuator : Actuator
lsref RangeActuator : Actuator

Formal Parameters

Sort *ENV_DOMAIN*

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

Postreferences : lsref Actuator : noMalAct

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.

Postreferences : lsref Actuator : noMalAct

Signature

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

Intention	: This object represents the part of an actuator describing the behavior of an actuator if it has no malfunction, i.e., if the machine assumes that it has no malfunction.
Postreferences	: dkref ToggleActuator : D_TA1 dkref ToggleActuator : D_TA2 dkref Pulse : D_P1 dkref RangeActuator : D_RA1 dkref Dimmer : D_D1 dkref ControlSystemActive : D_CSA1 dkref IndicatorLight : D_IL1 msref AreaCtrlPanelFM : I_M_ACPFM3 msref RoomCtrlPanel : I_M_RCP11 msref InformPerson : M_IP2 msref InformPerson : M_IP1 msref InformPerson : M_IP3 msref AreaCtrlPanelFM : I_M_ACPFM4 msref RoomCtrlPanel : I_M_RCP12 dkref InformPerson : D_IP1 msref InformPerson : I_M_IP2 dkref CeilingLightGroup : D_CLG1 dkref CeilingLightGroup : D_CLG3 msref CeilingLightGroup : M_CLG1 msref CeilingLightGroup : M_CLG2 msref CeilingLightGroup : M_CLG3 msref CeilingLightGroup : M_CLG4 dkref DimmableCeilingLightGroup : D_DCLG1 msref Area : M_A5 msref Area : M_A6 msref Area : M_A8 msref HallwaySection : M_HS5 msref Room : M_R8 msref Room : M_R8b msref Room : M_R9 msref Room : M_R10

Object *malAct* : *MalfunctionActuator*

Intention	: This object represents the part of an actuator describing the behavior of an actuator if it has a malfunction, i.e., if the machine assumes that it has a malfunction.
Postreferences	: msref HallwaySection : M_HS4 msref Room : M_R13 msref Room : M_R14

Template Declaration 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 classified as domain knowledge. Otherwise, if an actuator does not exist, they have to be classified as optative statements.

Postreferences : lsref Pulse : ToggleActuator

Base Classes

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

Intention : All definitions of description class *Actuator* are inherited 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.

Prereferences : akref Table2

Propreferences : rsref Area : R_A2

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.

Prereferences : akref Table2

Template Declaration Pulse

Intention

This class provides all definitions to describe the *pulse* of a ceiling light group by which it can be switched on and off. This class is derived from the description class *ToggleActuator*. The following specific parameters of a pulse are instantiated:

- the reaction time of a pulse

Postreferences	: lsref CeilingLightGroup : pulse

Base Classes

Class *ToggleActuator*

Intention	: All definitions of description class *ToggleActuator* are inherited.

Domain Knowledge

Property *D_P1*

Formal	: $\Box(noMalAct.reactionTime = 10)$
NL	: The reaction time of a pulse of a ceiling light group is 10 milliseconds.
Prereferences	: akref Table2

Template Declaration 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 classified as domain knowledge. Otherwise, if an actuator does not exist, it has to be classified as an optative statement.

Postreferences	: lsref Dimmer : RangeActuator lsref BinaryActuator : RangeActuator

Formal Parameters

Sort *ENV_DOMAIN*

Intention	: This domain contains the possible values of the phenomenon existing in the real world in the case that an actuator has no malfunction.
Postreferences	: lsref RangeActuator : Actuator lsref RangeActuator : modifyReaction

Sort *ADJUSTED_DOMAIN*

Intention	: This domain contains the possible values that can be adjusted by the machine in the case that an actuator has no malfunction.
Postreferences	: lsref RangeActuator : Actuator lsref RangeActuator : modifyReaction

Base Classes

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

Intention	: All definitions of description class *Actuator* are inherited.

Signature

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

Intention	: This function represents the way in which a value adjusted by the machine represented by the timed function *adjustedEntity* is modified during the reaction of an actuator resulting in the value of the entity in the real world represented by the timed function *envEntity*.
Scope	: eh
Postreferences	: dkref RangeActuator : D_RA1 dkref BinaryActuator : D_BA1

Domain Knowledge

Property *D_RA1*

Formal	: $\Box(noMalAct.adjustedEntity \rhd^{modifyReaction}_{noMalAct.reactionTime} noMalAct.envEntity)$
NL	: The value of the phenomenon in the real world is always derived from the value adjusted by the machine using the function *modifyReaction*. A change of the value adjusted by the machine is always propagated within the time *reactionTime*.
Prereferences	: akref BD_18

Template Declaration Dimmer

Intention

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

- the domains of a dimmer
- the reaction time of a dimmer

If a dimmer is set to $p\%$ this means that a ceiling light group emits $p\%$ of its possible maximal light illuminance.

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

Postreferences : lsref DimmableCeilingLightGroup : dimmer

Base Classes

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

Intention	: All definitions of description class *RangeActuator* are inherited 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_D1*

Formal	: $\Box(noMalAct.reactionTime = 10)$
NL	: The reaction time of a *dimmer* is 10 milliseconds.
Prereferences	: akref Table2

Template Declaration 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 an 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 classified as domain knowledge. Otherwise, if an actuator does not exist, it has to be classified as an optative statement.

Postreferences	: lsref ControlSystemActive : BinaryActuator lsref IndicatorLight : BinaryActuator

Base Classes

Class $RangeActuator(\ ENV\_DOMAIN = BINARY,$
$ADJUSTED\_DOMAIN = BINARY\ )$

Intention	: All definitions of description class *RangeActuator* are inherited 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 an actuator, resulting in a value of an actuator in the real world (*envEntity*), is the identity.

Template Declaration ControlSystemActive

Intention

This class provides all definitions to describe the signal *control system active* that can be sent by the machine in order to indicate that the machine is still active. This class is derived from the description class *BinaryActuator*. The following specific parameters of a *control system active* are instantiated :

- the reaction time of a *control system active*

If a*control system active* is set to 1 this means that the control system is still active.

Postreferences : lsref CeilingLightGroup : csa

Base Classes

Class *BinaryActuator*

Intention	: All definitions of description class *BinaryActuator* are inherited.

Domain Knowledge

Property *D_CSA1*

Formal	: $\Box(noMalAct.reactionTime = 10)$
NL	: The reaction time of a signal *control system active* is 10 ms.
Prereferences	: akref Table2

Template Declaration IndicatorLight

Intention

This class provides all definitions to describe an indicator light installed at a room control panel or a facility manger control panel. An indicator light indicates either a malfunction of a sensor or an actuator or an unreasonable input that has been done. This class is derived from the description class *BinaryActuator*. The following specific parameters of an indicator light are instantiated :

- the reaction time of an indicator light

If an *indicator light* is set to 1 this means that a light is on on a control panel. If it is set to 0 this means that this light on the control panel is off.

Postreferences : lsref InformPerson : indicator

Base Classes

Class *BinaryActuator*

Intention	: All definitions of description class *BinaryActuator* are inherited.

Domain Knowledge

Property *D_IL1*

Formal	: $\Box(noMalAct.reactionTime = 10)$
NL	: The reaction time of an *indicator light* is 10 ms.

Template Declaration PushButton

Intention

This class provides all definitions concerning the control of a ceiling light group using push buttons. Note that a push button is not visible for the machine and therefore this description class is not derived from the description class *Actuator*.

The value of the time constant *T_PB* has still to be provided by the domain expert.

Postreferences : lsref CeilingLightGroup : pb

Signature

Time Constant *T_PB*

NL	: push-to-light delay
Intention	: This time constant represents the delay between the time point a push button is pushed and the time point a ceiling light group changes to on or off.
Scope	: ev
Prereferences	: akref BD_8 akref BD_14
Postreferences	: dkref PushButton : D_PB1 dkref PushButton : D_PB2

Timed Predicate *light*

Intention	: This predicate represents the status of the ceiling light group existing in the real world. The predicate is true if and only if the ceiling light group is on. Note that *on* here means that the circuit of the ceiling light group is closed.
Scope	: eh
Prereferences	: akref BD_8 akref BD_14
Postreferences	: dkref PushButton : D_PB1 dkref PushButton : D_PB2 dkref CeilingLightGroup : D_CLG2

Timed Predicate *pushBut*

Intention	: This predicate represents the status of a push button existing in the real world. The predicate is true if and only if a push button is pushed.
Scope	: eh
Prereferences	: akref BD_8 akref BD_14
Postreferences	: dkref PushButton : D_PB1 dkref PushButton : D_PB2

Domain Knowledge

Property *D_PB1*

Formal	: $\Box(\neg light \wedge [pushBut] \rightarrow \Diamond_{\leq T\_PB} light)$
NL	: Whenever the ceiling light group is off and a push button is pressed, the ceiling light group changes to on within time T_PB.
Prereferences	: akref BD_8 akref BD_14

Property *D_PB2*

Formal	: $\Box(light \wedge [pushBut] \rightarrow \Diamond_{\leq T\_PB} \neg light)$
NL	: Whenever the ceiling light group is on and a push button is pressed, the ceiling light group changes to off within time T_PB.
Prereferences	: akref BD_8 akref BD_14

Group CombinedSensorsAndActuators

Intention

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

This are the following description classes:

- *Door*
- *CeilingLightGroup*
- *DimmableCeilingLightGroup*

Template Declaration Door

Intention

This class provides all definitions to describe a door. A door contains one door closed contact.

Postreferences :	lsref HallwaySection : eastDoor
	lsref HallwaySection : westDoor
	lsref Room : hallwayDoor
	lsref NeighborRoomConnections : neighborDoor

Signature

Object *doorClosedContact* : *DoorClosedContact*

NL :	door closed contact
Intention :	This object represents the closed contact of a door.
Prereferences :	akref BD_7
	akref BD_13
	akref Table1
Postreferences :	msref OfficeNeighbor : M_ON1
	msref NoOfficeNeighbor : M_NON1
	msref HallwaySection : M_HS4
	msref Room : M_R13
	msref Room : M_R14

Template Declaration CeilingLightGroup

Intention

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

- a status line indicating whether the ceiling light group is on or off represented by the object *ls*
- an actuator allowing the machine to switch on and off the ceiling light group, represented by the object *pulse*
- an actuator allowing the machine to signal that it is still active, represented by the object *csa*
- *numPB* many push buttons allowing a person to switch on and off the ceiling light group, represented by the object-array *pb*.
  Note that a push of a push button is not directly visible for the machine but only through the status line.

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

Postreferences :	lsref DimmableCeilingLightGroup : CeilingLightGroup lsref Area : light

Formal Parameters

Function $numPB \rightarrow NAT$

Intention :	This parameter represents the number of push buttons with direct control of the ceiling light group (bypassing the machine). The default number is 1, i.e., if nothing else is specified there is one such push button.
Default Value :	1
Postreferences :	lsref CeilingLightGroup : pb dkref CeilingLightGroup : D_CLG2 lsref Area : light

Signature

Object *sl* : *StatusLine*

Intention	: This object represents the status line indicating whether a ceiling light group is on or off.
Prereferences	: akref BD_8 akref BD_14 akref Table1 akref BD_19 akref Figure3
Postreferences	: dkref CeilingLightGroup : D_CLG1 dkref CeilingLightGroup : D_CLG2 dkref CeilingLightGroup : D_CLG3 msref CeilingLightGroup : M_CLG1 rsref Floor : R_F1 rsref Area : R_A1 rsref Area : R_A2 msref Area : M_A5 msref Area : M_A6 rsref HallwaySection : R_HS1 rsref HallwaySection : R_HS2 msref HallwaySection : M_HS4 msref Room : M_R13 msref Room : M_R14 msref Room : M_R8 msref Room : M_R8b msref Room : M_R9

Object *pulse* : *Pulse*

Intention	: This object represents the actuator of a ceiling light group the machine uses to switch the ceiling light group on or off.
Prereferences	: akref Table1 akref BD_19 akref Figure3
Postreferences	: dkref CeilingLightGroup : D_CLG1 msref CeilingLightGroup : M_CLG1 msref CeilingLightGroup : M_CLG2 msref Area : M_A5 msref Area : M_A6 msref Area : M_A8 msref HallwaySection : M_HS4 msref HallwaySection : M_HS5 msref Room : M_R8 msref Room : M_R8b msref Room : M_R9 msref Room : M_R10 msref Room : M_R13 msref Room : M_R14

Object *csa* : *ControlSystemActive*

Intention	: This object represents the actuator of a ceiling light group the machine uses to signal that it is still active.
Prereferences	: akref Table2 akref BD_19 akref Figure3
Postreferences	: dkref CeilingLightGroup : D_CLG3 msref CeilingLightGroup : M_CLG3 msref CeilingLightGroup : M_CLG4 dkref DimmableCeilingLightGroup : D_DCLG1 msref HallwaySection : M_HS4 msref Room : M_R13 msref Room : M_R14

Object *pb* : *PushButton*[*numPB*]

Intention	: These objects represent the push buttons of a ceiling light group a person can use to switch the light on or off. Note that a push of a button is not directly visible to the machine.
Prereferences	: akref BD_8 akref BD_14 akref BD_19 akref Figure3
Postreferences	: dkref CeilingLightGroup : D_CLG2

Timed Predicate *manualControl*

Intention	: This predicate is true if and only if the machine assumes that the ceiling light group is controlled manually.
Scope	: mh
Prereferences	: pdref PD_30
Postreferences	: msref CeilingLightGroup : I_CLG1 msref CeilingLightGroup : M_CLG1 msref CeilingLightGroup : M_CLG2 msref Area : M_A7

Domain Knowledge

Property *D_CLG1*

Formal	: $\Box(sl.noMalSens.envEntity = pulse.noMalAct.envEntity)$
NL	: The phenomenon in the real world which is measured by the status line is identical with the phenomenon adjusted by the pulse actuator.
Propreferences	: rsref Area : R_A2

Property *D_CLG2*

Formal	: $\Box(\forall i \in NAT(numPB) : ((sl.noMalSens.envEntity = 1) \leftrightarrow pb[i].light))$
NL	: The phenomenon in the real world which is measured by the status line is identical with the phenomenon adjusted by the push buttons.

Property *D_CLG3*

Formal	: $\Box(\blacksquare_{\geq 60000} csa.noMalAct.envEntity \neq 1 \rightarrow sl.noMalSens.envEntity = 1)$
NL	: If the signal *control system active* is not sent for at least 60 seconds, a ceiling light group is on.

Machine Specifications

Property *I_CLG1*

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

Property *M_CLG1*

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

Property *M_CLG2*

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 a ceiling light group is not controlled manually.

Property *M_CLG3*

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

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

Property *M_CLG4*

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

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

Template Declaration DimmableCeilingLightGroup

Intention

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

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

Postreferences : lsref Room : light

Base Classes

Class *CeilingLightGroup*($numPB = 1$)

Intention	: All definitions of description class *CeilingLightGroup* are inherited.
Postreferences	: lsref Room : light

Signature

Object *dimmer* : *Dimmer*

Intention	: This object represents the the dimmer the machine uses to set the illuminance of a ceiling light group.
Prereferences	: akref BD_13 akref BD_19 akref Figure3
Postreferences	: dkref DimmableCeilingLightGroup : D_DCLG1 msref Room : M_R8 msref Room : M_R8b msref Room : M_R9 msref Room : M_R10 msref Room : M_R13 msref Room : M_R14

Domain Knowledge

Property *D_DCLG1*

Formal	: $\square(\blacksquare_{\geq 60000} csa.noMalAct.envEntity \neq 1 \rightarrow dimmer.noMalAct.envEntity = 100)$
NL	: If the signal *control system active* is not sent for at least 60 seconds, the dimmer of a ceiling light group is set to 100%.

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*.

This are the following description classes:

- *InformPerson*
- *InformUnreasonableInput*
- *InformSensActMalFct*
- *AreaCtrlPanelFM*
- *RoomCtrlPanel*

Template Declaration InformPerson

Intention

This class provides all specifications concerning the informing of a person if a specific condition is given.

Postreferences :	lsref InformUnreasonableInput : InformPerson lsref InformSensActMalFct : InformPerson

Signature

Object *indicator* : *IndicatorLight*

Intention :	This object represents an indicator light on a control panel that is used to inform a person about a specific situation.
Prereferences :	pdref U8 pdref FM7 pdref NF9
Postreferences :	msref InformPerson : M_IP2 msref InformPerson : M_IP1 msref InformPerson : M_IP3 msref AreaCtrlPanelFM : I_M_ACPFM3 msref AreaCtrlPanelFM : I_M_ACPFM4 msref RoomCtrlPanel : I_M_RCP11 msref RoomCtrlPanel : I_M_RCP12 dkref InformPerson : D_IP1 msref InformPerson : I_M_IP2 msref Floor : M_F5

Time Constant *T_MinInf*

Intention :	This time constant is the minimal time a person has to be informed about a specific situation.
Scope :	mh
Prereferences :	pdref U8 pdref FM7 pdref NF9
Postreferences :	msref InformPerson : M_IP2 msref InformPerson : M_IP4 msref InformUnreasonableInput : M_IUI1 msref InformSensActMalFct : M_ISAMF1

Time Constant *T_MaxInf*

Intention	: This time constant is the maximal time a person has to be informed about a specific situation.
Scope	: mh
Prereferences	: pdref U8 pdref FM7 pdref NF9
Postreferences	: msref InformPerson : M_IP3 msref InformPerson : M_IP4 msref InformUnreasonableInput : M_IUI2 msref InformSensActMalFct : M_ISAMF2

Time Constant *T_InfDelayNew*

Intention	: This time constant represents the maximal delay between the time point a specific situation about a person should be informed occurs and the time point a person is informed about this situation.
Scope	: mh
Prereferences	: pdref U8 pdref FM7 pdref NF9
Postreferences	: msref InformPerson : M_IP1 msref InformUnreasonableInput : M_IUI3 msref InformSensActMalFct : M_ISAMF3

Timed Predicate *informCondEnv*

Intention	: This predicate is true if and only if there is a situation a person should be informed about. This predicate represent the situation as it is in the real world.
Scope	: eh
Prereferences	: pdref U8 pdref FM7 pdref NF9

Timed Predicate *informCondMach*

Intention	: This predicate is true if and only if there is a situation a person should be informed about.
Scope	: ev
Prereferences	: pdref U8 pdref FM7 pdref NF9
Postreferences	: msref InformPerson : I_M_IP1 msref InformPerson : M_IP1 msref AreaCtrlPanelFM : M_ACPFM1 msref RoomCtrlPanel : M_RCP1 msref Floor : M_F3 msref Floor : M_F4 msref HallwaySection : M_HS4 msref Room : M_R13 msref Room : M_R14

Timed Predicate *personInformed*

Intention	: This predicate is true if and only if a person is informed.
Scope	: eh
Prereferences	: pdref U8 pdref FM7 pdref NF9
Postreferences	: dkref InformPerson : D_IP1

Domain Knowledge

Property *D_IP1*

Formal	: $\Box(indicator.noMalAct.envEntity = 1 \rightarrow personInformed)$
NL	: Whenever the indicator light on a control panel is on then the corresponding person is informed.

Machine Specifications

Property *I_M_IP1*

Formal	: $\neg informCondMach$
NL	: Initially the situation a person has to be informed about is not given.

Property *I_M_IP2*

Formal	: $indicator.noMalAct.adjustedEntity = 0$
NL	: Initially the indicator light is switched off.

Property *M_IP1*

Formal	: $\Box([informCondMach] \rightarrow \Diamond_{\leq T\_InfDelayNew}\, indicator.noMalAct.adjustedEntity = 1)$
NL	: Whenever the situation a person should be informed about starts the indicator light on the control panel is on within *T_InfDelayNew* milliseconds.

Property *M_IP2*

Formal : $\Box(\nabla_{\leq T\_MinInf}(indicator.noMalAct.adjustedEntity = 1))$

NL : Whenever the indicator light is switched on it remains on for at least *T_MinInf* milliseconds.

Property *M_IP3*

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

NL : Whenever the indicator light is switched on it is switched off again within the next *T_MaxInf* milliseconds.

Property *M_IP4*

Formal : $\Box(T\_MinInf < T\_MaxInf)$

NL : The maximal time a person has to be informed about a specific situation is always greater than the minimal time a person has to be informed about a specific situation.

Template Declaration InformUnreasonableInput

Intention

This class provides all specifications concerning the informing of a person about an unreasonable input. This class is derived from the description class *InformPerson*. The following specific parameters are instantiated here:

- the minimal time a person has to be informed about an unreasonable input
- the maximal time a person has to be informed about an unreasonable input

Postreferences	: lsref AreaCtrlPanelFM : indicatorUnreasonInp lsref RoomCtrlPanel : indicatorUnreasonInp

Base Classes

Class *InformPerson*

Intention	: All definitions of description class *InformPerson* are inherited.

Machine Specifications

Property *M_IUI1*

Formal	: $\Box(T\_MinInf = 1000)$
NL	: The minimal time a person is informed about an unreasonable input is 1 second.

Property *M_IUI2*

Formal	: $\Box(T\_MaxInf = 5000)$
NL	: The maximal time a person is informed about an unreasonable input is 5 second.

Property *M_IUI3*

Formal	: $\Box(T\_InfDelayNew = 100)$
NL	: The maximal delay between the time point a person makes an unreasonable input and the time point the indicator light is on is 100 milliseconds.

Template Declaration InformSensActMalFct

Intention

This class provides all specifications concerning the informing of a person about a malfunction of a sensor or actuator. This class is derived from the description class *InformPerson*. The following specific parameters are instantiated here:

- the minimal time a person has to be informed about such a malfunction
- the maximal time a person has to be informed about such a malfunction

Postreferences : lsref AreaCtrlPanelFM : indicatorMalFct
lsref RoomCtrlPanel : indicatorMalFct
lsref Floor : indicatorMalFctWestStaircase
lsref Floor : indicatorMalFctEastStaircase

Base Classes

Class *InformPerson*

Intention : All definitions of description class *InformPerson* are inherited.

Machine Specifications

Property *M_ISAMF1*

Formal : $\Box(T\_MinInf = 10000)$
NL : The minimal time a person is informed about a malfunction of a sensor or an actuator is 10 seconds.

Property *M_ISAMF2*

Formal : $\Box(T\_MaxInf = 86400000)$
NL : The maximal time a person is informed about a malfunction of a sensor or an actuator is 24 hours.

Property *M_ISAMF3*

Formal : $\Box(T\_InfDelayNew = 100)$
NL : The maximal delay between the time point a sensor or an actuator has an malfunction and the time point the indicator light is on is 100 milliseconds.

Template Declaration AreaCtrlPanelFM

Intention

This class provides all specifications concerning the *facility manager control panel*, i.e., all possibilities the facility manager has to control the light in an area using his control panel. An area is here the generalization of rooms and hallways.

These possibilities are:

- a possibility to switch off all ceiling light groups in an area
- a possibility to determine the time an area has to be unoccupied continuously before the ceiling light groups are switched off.

Postreferences : lsref Area : CPFM

Signature

Object *lightOff* : *EnteredValue*(*VALID_INPUTS = BINARY*)

Intention : This object represents whether the facility manager has switched off all ceiling light groups in an area.

- *lightOff.checked* = 1 means that the ceiling light groups are switched off by the facility manager;
- *lightOff.checked* = 0 means that the ceiling light groups are not switched off by the facility manager.
  Note that this does not imply that the ceiling light groups are on!

Prereferences : pdref FM6
Postreferences : msref AreaCtrlPanelFM : I_M_ACPFM2
msref AreaCtrlPanelFM : M_ACPFM1
rsref Area : R_A1

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

Intention : This object represents the time an area has to be unoccupied continuously before the lights in this area are switched off.
Prereferences : pdref FM2
pdref FM3
pdref FM4
pdref FM5
Postreferences : msref AreaCtrlPanelFM : I_M_ACPFM1
msref AreaCtrlPanelFM : M_ACPFM1
msref Area : M_A3

Object *indicatorUnreasonInp* : *InformUnreasonableInput*

Intention	: This object represents the handling of an unreasonable input.
Prereferences	: pdref NF9
Postreferences	: msref AreaCtrlPanelFM : I_M_ACPFM3 msref AreaCtrlPanelFM : M_ACPFM1 msref Floor : M_F5

Object *indicatorMalFct* : *InformSensActMalFct*

Intention	: This object represents the handling of a malfunction of a sensor or an actuator.
Prereferences	: pdref FM7
Postreferences	: msref AreaCtrlPanelFM : I_M_ACPFM4 msref HallwaySection : M_HS4 msref Room : M_R14

Machine Specifications

Property *I_M_ACPFM1*

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_M_ACPFM2*

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

Property *I_M_ACPFM3*

Formal	: $indicatorUnreasonInp.indicator.noMalAct.adjustedEntity = 0$
NL	: Initially the indicator light on the facility manager control panel concerning unreasonable inputs is off.

Property *I_M_ACPFM4*

Formal	: $indicatorMalFct.indicator.noMalAct.adjustedEntity = 0$
NL	: Initially the indicator light on the facility manager control panel concerning malfunctions of sensors or actuators is off.

Property *M_ACPFM1*

Formal : $\Box(\mathit{indicatorUnreasonInp.informCondMach} \leftrightarrow \mathit{lightOff.unreasonableInput} \vee \mathit{unoccDelay.unreasonableInput})$

NL : The situation a facility manager has to be informed about concerning an unreasonable input is always given if and only if there is an unreasonable input concerning one of the following inputs:

- the switching off of the ceiling light groups in an area
- the determination of the time an area has to be unoccupied continuously before the ceiling light groups in this area are switched off.

Prereferences : pdref FM4
pdref FM5
pdref FM6

Template Declaration RoomCtrlPanel

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

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

- A chosen light scene has to be established. This is the case if the value of *lightSceneSwitch.checked* is 1.
- The two ceiling light groups are set directly using the control panel. This is the case if the value of *lightSceneSwitch.checked* is 0.

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

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

Postreferences : lsref Room : CPRoom

Signature

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

NL	: control panel light scene
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*.
Prereferences	: pdref U5
Postreferences	: msref Room : M_R7
	msref Room : M_R5
	msref RoomCtrlPanel : I_M_RCP1
	msref RoomCtrlPanel : M_RCP1

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

NL : window ceiling light switch of the control panel

Intention : This object represents the value defined for the window ceiling light group by using the control panel.

- *winCeilLightSwitch.checked* = 1 means that the window ceiling light group should be on.
- *winCeilLightSwitch.checked* = 0 means that the window ceiling light group should be off.

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

Prereferences : pdref U9

Postreferences : msref RoomCtrlPanel : I_M_RCP2
msref RoomCtrlPanel : M_RCP1
msref Room : M_R9

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

NL : wall ceiling light switch of the control panel

Intention : This object represents the value defined for the wall ceiling light group by using the control panel.

- *wallCeilLightSwitch.checked* = 1 means that the wall ceiling light group should be on.
- *wallCeilLightSwitch.checked* = 0 means that the wall ceiling light group should be off.

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

Prereferences : pdref U9

Postreferences : msref RoomCtrlPanel : I_M_RCP3
msref RoomCtrlPanel : M_RCP1
msref Room : M_R9

Object *winCeilLightDimmer* : *EnteredValue*(*VALID_INPUTS = PERCENT*)

NL : window ceiling light dimmer of the control panel

Intention : This object represents the value defined for the dimmer of the window ceiling light group by using the control panel. If a dimmer is set to $p\%$ this means that a ceiling light group emits $p\%$ of its possible maximal light illuminance.
Note that this setting is only relevant if the ceiling light group is not controlled manually using the push buttons.

Prereferences : pdref U9

Postreferences : msref RoomCtrlPanel : I_M_RCP4
msref RoomCtrlPanel : M_RCP1

Object *wallCeilLightDimmer* : *EnteredValue*(*VALID_INPUTS* = *PERCENT*)

NL	: wall ceiling light dimmer of the control panel
Intention	: This object represents the value defined for the dimmer of the wall ceiling light group by using the control panel. If a dimmer is set to p% this means that a ceiling light group emits p% of its possible maximal light illuminance. Note that this setting is only relevant if the ceiling light group is not controlled manually using the push buttons.
Prereferences	: pdref U9
Postreferences	: msref RoomCtrlPanel : I_M_RCP5 msref RoomCtrlPanel : M_RCP1

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

NL : light scenes are enabled by the control panel

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 ceiling light switches (*winCeilLightSwitch*, *winCeilLightSwitch*), and the dimmer settings (*winCeilLightDimmer*, *wallCeilLightSwitch*) should be applied (this is the case for *lightSceneSwitch* = 0).

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

Postreferences : msref Room : M_R3
msref RoomCtrlPanel : I_M_RCP6
msref RoomCtrlPanel : M_RCP1
msref Room : M_R8
msref Room : M_R8b
msref Room : M_R9

Object *reoccDelay* : *EnteredValue*(*VALID_INPUTS* = *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 chosen light scene.
Prereferences	: pdref U7 pdref U9
Postreferences	: msref RoomCtrlPanel : I_M_RCP7 msref Room : M_R5 msref Room : M_R6 msref RoomCtrlPanel : M_RCP1 msref Room : M_R4

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

Intention	: This object represents the default light scene that can be determined for a room using the room control panel.
Prereferences	: pdref U6
Postreferences	: msref Room : M_R2 msref RoomCtrlPanel : I_M_RCP8 msref RoomCtrlPanel : I_M_RCP9 msref RoomCtrlPanel : I_M_RCP10 msref RoomCtrlPanel : M_RCP1

Object *indicatorUnreasonInp* : *InformUnreasonableInput*

Intention	: This object represents the handling of an unreasonable input.
Prereferences	: pdref NF9
Postreferences	: msref RoomCtrlPanel : I_M_RCP11 msref RoomCtrlPanel : M_RCP1

Object *indicatorMalFct* : *InformSensActMalFct*

Intention	: This object represents the handling of a malfunction of a sensor or an actuator.
Prereferences	: pdref U8
Postreferences	: msref RoomCtrlPanel : I_M_RCP12 msref Room : M_R13

Requirement Specifications

Property *R_RCP1*

NonFormal	: The control panel should be easy and intuitive to use.
Prereferences	: pdref NF8

Machine Specifications

Property *I_M_RCP1*

Formal	: $chosenLightScene.checked$ = "DEFAULT"
NL	: Initially the default light scene is also the chosen light scene.

Property *I_M_RCP2*

Formal	: $winCeilLightSwitch.checked = 1$
NL	: Initially the window ceiling light group should be on according to the control panel.

Property *I_M_RCP3*

Formal	: $wallCeilLightSwitch.checked = 1$
NL	: Initially the wall ceiling light group should be on according to the control panel.

Property *I_M_RCP4*

Formal	: $winCeilLightDimmer.checked = 100$
NL	: Initially the dimmer of the window ceiling light group is set to 100% according to the control panel.

Property *I_M_RCP5*

Formal : $wallCeilLightDimmer.checked = 100$
NL : Initially the dimmer of the wall ceiling light group is set to 100% according to the control panel.

Property *I_M_RCP6*

Formal : $lightSceneSwitch.checked = 1$
NL : Initially the light scenes are enabled by the control panel.

Property *I_M_RCP7*

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 *I_M_RCP8*

Formal : $defaultLightScene.checked.name =$ "DEFAULT"
NL : Initially the name of the default light scene is *DEFAULT*.

Property *I_M_RCP9*

Formal : $defaultLightScene.checked.ambLightLevel = 1000$
NL : Initially the desired ambient light level of the default light scene is 1000 lux.

Property *I_M_RCP10*

Formal : $defaultLightScene.checked.option = LightScene :: BOTH$
NL : Initially the option of the default light scene is *BOTH*

Property *I_M_RCP11*

Formal : $indicatorUnreasonInp.indicator.noMalAct.adjustedEntity = 0$
NL : Initially the indicator light on the room control panel concerning unreasonable inputs is off.

Property *I_M_RCP12*

Formal : $indicatorMalFct.indicator.noMalAct.adjustedEntity = 0$
NL : Initially the indicator light on the room control panel concerning malfunctions of sensors or actuators is off.

Property *M_RCP1*

Formal : $\Box(\mathit{indicatorUnreasonInp.informCondMach} \leftrightarrow$
$\mathit{chosenLightScene.unreasonableInput} \vee$
$\mathit{winCeilLightSwitch.unreasonableInput} \vee$
$\mathit{wallCeilLightSwitch.unreasonableInput} \vee$
$\mathit{winCeilLightDimmer.unreasonableInput} \vee$
$\mathit{wallCeilLightDimmer.unreasonableInput} \vee$
$\mathit{lightSceneSwitch.unreasonableInput} \vee$
$\mathit{reoccDelay.unreasonableInput} \vee$
$\mathit{defaultLightScene.unreasonableInput})$

NL : The situation a user has to be informed about concerning an unreasonable input is always given if and only if there is an unreasonable input concerning one of the following inputs:

- the currently chosen light scene
- the switching on or off of the window ceiling light group
- the switching on or off of the wall ceiling light group
- the setting of the dimmer of the window ceiling light group
- the setting of the dimmer of the wall ceiling light group
- the determination whether light scenes are enabled or not
- the determination of the time within the reoccupation of a room leads to a reestablishing of the previous chosen light scene
- the setting of the default light scene

Prereferences : pdref U5
pdref U6
pdref U7
pdref U9

Group Occupancy

Intention

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

This are the following description classes:

- *NoMalfunctionOccupancy*
- *MalfunctionOccupancy*
- *Occupancy*

Template Declaration 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

- a time value *delay* and
- on a predicate indicating whether a person is currently in an 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.

Postreferences : lsref Occupancy : noMalOcc

Signature

Timed Function $delay \rightarrow TIME$

Intention	: This function represents the maximal time an area can be empty – after it has been not empty – but still being defined to be occupied.
Scope	: mh
Prereferences	: pdref U3 pdref U4 pdref U7 pdref U9 pdref FM2 pdref FM3 pdref FM4 pdref FM5
Postreferences	: msref NoMalfunctionOccupancy : M_NMO1 msref Room : M_R4 msref Area : M_A3

Timed Predicate *areaEmpty*

NL	: =1 : the machine supposes an empty area. =0 : the machine supposes a non-empty area
Intention	: This predicate is true if and only if the machine assumes that there is currently no person in the area for which occupancy is considered.
Scope	: mh
Prereferences	: pdref U3 pdref U4 pdref U7 pdref U9 pdref FM2 pdref FM3 pdref FM4 pdref FM5
Postreferences	: msref NoMalfunctionOccupancy : M_NMO1 msref HallwaySection : M_HS3 msref Room : M_R11 msref Room : M_R5 msref Room : M_R6 msref Room : M_R12

Timed Predicate *occupied*

NL	: =1 : the machine supposes an occupied area =0 : the machine supposes an unoccupied area
Intention	: This predicate is true if and only if the machine assumes that the area for which occupancy is considered is occupied under the precondition that there is no malfunction of the sensors used to determine occupancy (compare timed predicate *occUsed* in description class *Occupancy*). 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 an area or not. See timed predicate *areaEmpty*.
Scope	: mh
Prereferences	: pdref U3 pdref U4 pdref U7 pdref U9 pdref FM2 pdref FM3 pdref FM4 pdref FM5
Postreferences	: msref NoMalfunctionOccupancy : M_NMO1 msref Occupancy : M_O1

Machine Specifications

Property *M_NMO1*

Formal	: $\Box(occupied \leftrightarrow \blacklozenge_{\leq delay} \neg areaEmpty)$
NL	: An area is defined to be occupied if and only if during the past *delay* milliseconds the area has been not empty continuously, i.e., during this time a person has been in the area.
Prereferences	: pdref U3 pdref U4 pdref FM2 pdref FM3
Propreferences	: rsref Area : R_A2

Template Declaration 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 refined in the description classes where they are used.

Postreferences : lsref Occupancy : malOcc

Signature

Object *manualEnteredOcc* : *EnteredValue*(*VALID_INPUTS* = *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.

- $manualEnteredOcc.checked = 1$ means occupancy,
- $manualEnteredOcc.checked = 0$ means no occupancy.

Note that the value of this object is only meaningful if the predicate *stateManualEnteredOcc* is true.

Prereferences : pdref FM11

Postreferences : msref MalfunctionOccupancy : M_MO2
msref MalfunctionOccupancy : I_M_MO1

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

Intention : Manually entered value concerning a malfunction of the sensors determining occupancy entered by a user.

- 1 means that some sensors used to determine occupancy have a malfunction,
- 0 means that no sensor used to determine occupancy has a malfunction.

Note that the value of this object is only meaningful if the timed predicate *stateManualEnteredMal* is true.

Prereferences : pdref FM11

Postreferences : msref MalfunctionOccupancy : M_MO1
msref MalfunctionOccupancy : I_M_MO5

Timed Predicate *malOccDetEnv*

Intention	: This predicate is true if and only if 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
Prereferences	: pdref NF4

Timed Predicate *malOccDetMach*

Intention	: This predicate is true if and only if 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
Prereferences	: pdref NF4
Postreferences	: msref Occupancy : M_O1 msref MalfunctionOccupancy : I_M_MO2 msref MalfunctionOccupancy : M_MO1

Timed Predicate *stateManualEnteredOcc*

Intention	: This predicate is true if and only if it has been entered manually whether there is occupancy or not.
Scope	: mh
Prereferences	: pdref FM11 pdref PD_30
Postreferences	: msref MalfunctionOccupancy : I_M_MO3 msref MalfunctionOccupancy : M_MO2

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
Prereferences	: pdref NF4 pdref PD_30
Postreferences	: msref MalfunctionOccupancy : M_MO2 msref Area : M_A4 msref MalfunctionOccupancy : I_M_MO4

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
Prereferences	: pdref NF4 pdref PD_30
Postreferences	: msref MalfunctionOccupancy : M_MO2 msref Occupancy : M_O1

Timed Predicate *stateManualEnteredMal*

Intention	: This predicate is true if and only if it has been manually entered whether there is a malfunction of the sensors used to determine occupancy or whether there is no malfunction of them.
Scope	: mh
Prereferences	: pdref FM11
Postreferences	: msref MalfunctionOccupancy : M_MO1 msref MalfunctionOccupancy : I_M_MO6

Timed Predicate *automaticDetMal*

Intention	: This predicate is true if and only if a malfunction of the sensors used to determine occupancy has been detected automatically, i.e. not by entering it manually.
Scope	: mh
Prereferences	: pdref FM11
Postreferences	: msref MalfunctionOccupancy : M_MO1 msref MalfunctionOccupancy : I_M_MO7

Machine Specifications

Property *I_M_MO1*

Formal	: $manualEnteredOcc.checked = 0$
NL	: Initially the value concerning occupancy entered manually is 0.

Property *I_M_MO2*

Formal	: $\neg malOccDetMach$
NL	: Initially the machine does not assume that some sensors used to determine occupancy have a malfunction.

Property *I_M_MO3*

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

Property *I_M_MO4*

Formal : $\neg automaticDetermValue$
NL : Initially the value for the automatically determined occupancy in the case of a malfunction of the sensors used to determine occupancy is false.

Property *I_M_MO5*

Formal : $manualEnteredMal.checked = 0$
NL : Initially the value of the manually entered value concerning a malfunction of the sensors used to determine occupancy is 0.

Property *I_M_MO6*

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

Property *I_M_MO7*

Formal : $\neg automaticDetMal$
NL : Initially no malfunction of the sensors used to determine occupancy is detected automatically.

Property *M_MO1*

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

NL : The machine assumes a malfunction of some sensors used to determine occupancy if and only if 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.

Prereferences : pdref FM11
pdref PD_30

Property *M_MO2*

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.
Prereferences	: pdref PD_30

Template Declaration Occupancy

Intention

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

- occupancy is specified under the condition that all sensors used to determine occupancy have no malfunction
- occupancy is specified under the condition that some sensors used to determine occupancy 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.

Postreferences : lsref Area : occFM
lsref Room : occRoom

Signature

Object *noMalOcc* : *NoMalfunctionOccupancy*

Intention : This object represents the part of the occupancy determination describing the behavior if the sensors used to determine occupancy have no malfunction, i.e., if the machine assumes that they have no malfunction.

Postreferences : msref Occupancy : M_O1
msref HallwaySection : M_HS3
msref Room : M_R11
msref Room : M_R5
msref Room : M_R4
msref Room : M_R6
msref Room : M_R12
msref Area : M_A3

Object *malOcc* : *MalfunctionOccupancy*

Intention : This object represents the part of the occupancy determination describing the behavior if the sensors used to determine occupancy have a malfunction, i.e., if the machine assumes that they have a malfunction.

Postreferences : msref Occupancy : M_O1
msref Area : M_A4

Timed Predicate *occUsed*

Intention : This predicate is true if and only if the machine assumes occupancy independent whether a sensor used to determine occupancy has a malfunction or not (compare timed predicate *occupied* in description class *NoMalfunctionOccupancy*).
Scope : mh
Prereferences : pdref PD_30
Postreferences : msref Occupancy : M_O1
msref Room : M_R7
rsref Area : R_A3
rsref Floor : R_F1
rsref Area : R_A1
msref Area : M_A5
msref Area : M_A6
msref Area : M_A7
msref Room : M_R8
msref Room : M_R8b
msref Room : M_R9

Timed Predicate *occEnv*

Intention : This predicate is true if and only if an area is occupied. Note that this predicate represents the phenomenon as it exists in the environment.
Scope : eh
Postreferences : dkref HallwaySection : D_HS1
rsref Area : R_A2

Machine Specifications

Property *M_O1*

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.

Prereferences : pdref PD_30
Propreferences: rsref Area : R_A2

Group BuildingUnits

Intention

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

This are the following description classes:

- *Area*
- *HallwaySection*
- *Room*
- *NeighborRoomConnections*
- *Office*
- *OfficeNeighbor*
- *OfficeNoNeighbor*
- *NoOffice*
- *NoOfficeNeighbor*
- *NoOfficeNoNeighbor*
- *Staircase*
- *Floor*

Template Declaration Area

Intention

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

- A set of ceiling light groups that can be controlled by the machine. The concrete number of controlled ceiling light groups 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 ceiling light groups using the facility manager control panel.
- The specification of the behavior of the system concerning the case that the ceiling light groups are controlled manually.

Postreferences	: lsref HallwaySection : Area lsref Room : Area

Formal Parameters

Function *numLights* $\to$ *NAT*

Intention	: This parameter represents the number of lights in an area that are controlled by the machine.
Postreferences	: lsref Area : light lsref Area : evalSaveLight rsref Area : R_A1 rsref Area : R_A2 msref Area : M_A5 msref Area : M_A6 msref Area : M_A7 msref Area : M_A8

Function *numPB* $\to$ *NAT*

Intention	: This parameter represents the number of push buttons that can be used by a person to control a ceiling light group in an area by-passing the machine. The default number is 1, i.e., if nothing else is specified there is one such push button.

Signature

Object *occFM* : *Occupancy*

Intention	: This object represents the part of the light control concerning occupancy in an area as it is seen by the facility manager. Note that in the case of rooms there is also a second version of the aspect occupancy concerning the handling of light scenes (see description class *Room*, object *occRoom*.
Prereferences	: pdref FM2 pdref FM3 pdref FM4 pdref FM5 pdref FM6
Postreferences	: msref Area : M_A4 msref HallwaySection : M_HS3 rsref Area : R_A3 msref Room : M_R11 dkref HallwaySection : D_HS1 rsref Floor : R_F1 rsref Area : R_A1 rsref Area : R_A2 msref Area : M_A3 msref Area : M_A5 msref Area : M_A6 msref Area : M_A7

Object *light* : *CeilingLightGroup*[*numLights*]

Intention	: These objects represent the ceiling light groups in an area that are controlled by the machine.
Prereferences	: akref BD_8 akref BD_14 akref BD_19
Postreferences	: lsref Room : light rsref Floor : R_F1 rsref Area : R_A1 rsref Area : R_A2 msref Area : M_A5 msref Area : M_A6 msref Area : M_A7 msref Area : M_A8 rsref HallwaySection : R_HS1 rsref HallwaySection : R_HS2 msref HallwaySection : M_HS4 msref HallwaySection : M_HS5

Object *CPFM* : *AreaCtrlPanelFM*

Intention	: This object represents the facility manager control panel containing the entities for which the facility manager can enter a value.
Prereferences	: pdref FM4 pdref FM5 pdref FM6
Postreferences	: rsref Area : R_A1 msref Area : M_A3 msref HallwaySection : M_HS4 msref Room : M_R14 msref Floor : M_F5

Time Constant *T_A1*

NL	: occupation-to-safelight delay
Intention	: This time constant represents the maximal time delay for the following situations: • between the time point the facility manager switches off all ceiling light groups in an unoccupied area and the time point all ceiling light groups in this area are off, see Property *R_A1*. • between the time point an area becomes unoccupied and the time point the machine controlled lights are off, see Property *R_A2*. • between the time point an area becomes occupied and the time point at which safe light is established, see Property *R_A3*. This time constant is used to state requirement specifications.
Scope	: ev
Prereferences	: pdref U1 pdref U13 pdref FM2 pdref FM3 pdref FM6
Postreferences	: msref Area : M_A1 msref Area : M_A9 rsref Area : R_A3 rsref Area : R_A1 rsref Area : R_A2 msref HallwaySection : M_HS5

Time Constant *T_A2*

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. This time constant is used to state machine specifications.
Scope	: mh
Prereferences	: pdref U1 pdref U13
Postreferences	: msref Area : M_A2 msref Area : M_A9 msref Area : M_A5 msref Area : M_A6 msref HallwaySection : M_HS5

Timed Function *dayLight* $\rightarrow$ *LIGHT_VALUES*

Intention	: This function represents the day light that the machine assumes to be available in an area.
Scope	: mh
Prereferences	: pdref FM1
Postreferences	: msref Area : M_A5 msref Room : M_R8 msref Room : M_R8b

Timed Predicate *safeLight*

NL	: safe light
Intention	: This predicate is true if and only if there is sufficient light to move safely.
Scope	: eh
Prereferences	: pdref U1 pdref U13
Postreferences	: rsref Area : R_A3

Timed Predicate *noSaveLight*

Intention	: This predicate is true if and only if 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 ceiling light groups are controlled manually.
Scope	: mh
Prereferences	: pdref U1
Postreferences	: msref Area : I_M_A1 msref HallwaySection : M_HS2 msref Room : M_R3 rsref Area : R_A3 msref Area : M_A5

Timed Predicate *manualControl*

Intention	: This predicate is true if and only if the machine assumes that the ceiling light groups light in this area are controlled manually.
Scope	: mh
Prereferences	: pdref PD_30
Postreferences	: msref HallwaySection : M_HS2 msref Room : M_R3 msref Area : M_A7 msref Area : M_A8 msref Room : M_R8 msref Room : M_R8b msref Room : M_R9 msref Room : M_R10

Timed Predicate *sensorActMalFct*

Intention	: This predicate is true if and only if a sensor or an actuator that is solely used by a room has a malfunction.
Scope	: mh
Prereferences	: pdref U8

Function *evalSaveLight* : $NAT(numLights)$, $LIGHT\_VALUES \rightarrow BINARY$

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]=1 means that the i–th light has to be on. This computation depends at least from the current day light that the machine assumes to be available in an area.
Scope	: mh
Prereferences	: pdref U1 pdref U13 pdref FM1
Postreferences	: msref Area : M_A5

Requirement Specifications

Property *R_A1*

Formal	: $\Box(CPFM.lightOff.checked = 1 \wedge \neg occFM.occUsed \Rightarrow_{\leq T\_A1}$ $\forall n \in NAT(numLights):$ $(light[n].sl.noMalSens.envEntity = 0))$
NL	: Whenever the light is switched off by the facility manager using the facility manager control panel and the area is unoccupied for at least *T_A_1* milliseconds, all ceiling light groups are off within this time and remain off at least as long as the preconditions are fulfilled.

Property *R_A2*

Formal	: $\Box(\neg occFM.occEnv \Rightarrow_{\leq T\_A1}$ $\forall n \in NAT(numLights):$ $(light[n].sl.noMalSens.envEntity = 0))$
NL	: Whenever an area is unoccupied for at least *T_A1* milliseconds, all ceiling light groups in this area are off within this time and remain off at least as long as the area is unoccupied.
Propreferences	: dkref HallwaySection : D_HS1 dkref ToggleActuator : D_TA1 msref Area : M_A6 msref HallwaySection : M_HS3 msref Occupancy : M_O1 msref NoMalfunctionOccupancy : M_NMO1 dkref CeilingLightGroup : D_CLG1 msref HallwaySection : M_HS5

Property *R_A3*

Formal	: $\Box(\neg noSaveLight \wedge occFM.occUsed \Rightarrow_{\leq T\_A1} safeLight)$
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.
Prereferences	: pdref U1 pdref U13

Machine Specifications

Property *I_M_A1*

Formal	: $\neg noSaveLight$
NL	: Initially safe light has to be established if an area is occupied.

Property *M_A1*

Formal	: $\Box(T\_A1 = 50)$
NL	: The value of *T_AL_1* is always 50 milliseconds.

Property *M_A2*

Formal	: $\Box(T\_A2 = 30)$
NL	: The value of *T_AL_2* is always 30 milliseconds.

Property *M_A3*

Formal	: $\Box(occFM.noMalOcc.delay = CPFM.unoccDelay.checked)$
NL	: The time delay used to determine occupancy as it is seen by the facility manager corresponds always with the value entered by the facility manager for the time an area has to be unoccupied continuously before the lights are switched off.

Property *M_A4*

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

NL : The value determined automatically by the machine for occupancy as it is seen by the facility manager in the case of a malfunction of some sensors used to determine occupancy is always true, i.e., occupancy is assumed.

Prereferences : pdref NF4

Property *M_A5*

Formal : $\Box(\neg noSaveLight \wedge occFM.occUsed \rightarrow \forall n \in NAT(numLights):$
$(evalSaveLight(n,\ dayLight) = 1 \wedge light[n].sl.usedValue = 0$
$\rightarrow \Diamond_{\leq T\_A2}\, light[n].pulse.noMalAct.adjustedEntity = 1))$

NL : Whenever safe light has to be established in an area and this area is occupied then for all ceiling light groups in this area it holds: if a ceiling light group has to be on in order to achieve safe light and this ceiling light group is off then within T_A2 milliseconds a pulse to this ceiling light group is sent.

Prereferences : pdref U1

Property *M_A6*

Formal : $\Box(\neg occFM.occUsed \rightarrow$
$\forall n \in NAT(numLights):$
$(light[n].sl.usedValue = 1$
$\rightarrow \Diamond_{\leq T\_A2}\, light[n].pulse.noMalAct.adjustedEntity = 1))$

NL : Whenever the machine assumes no occupancy of an area then for all ceiling light groups in the area it holds: if a ceiling light group is on then within T_A2 milliseconds a pulse is sent.

Prereferences : pdref FM3
pdref FM2

Propreferences : rsref Area : R_A2

Property *M_A7*

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

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

Property *M_A8*

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 ceiling light groups remain unchanged.

Property *M_A9*

Formal	: $\Box(T\_A2 < T\_A1)$
NL	: The time constant *T_AL_2* is always smaller than the time constant *T_AL_1*.

Template Declaration HallwaySection

Intention

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

- one ceiling light group
- three motion detectors: one at each end of a hallway section and one in the middle of a hallway section.

Postreferences	: lsref Floor : h1 lsref Floor : h2 lsref Floor : h3 lsref Floor : HALLWAYS

Formal Parameters

Function $numPB \rightarrow NAT$

Intention	: This parameter represents the number of push buttons that can be used by a person to control the ceiling light group in a hallway section by bypassing the machine.
Postreferences	: lsref HallwaySection : Area

Base Classes

Class $Area(\ numLights = 1,\ \ numPB = numPB\ )$

Intention	: All definitions of description class *Area* are inherited. There is one ceiling light group in a hallway section that is controlled by the machine and there are *numPB* many push buttons for controlling this ceiling light group.

Signature

Object *eastMd* : *MotionDetector*

NL	: motion detector
Intention	: This object represents the motion detector in a hallway section which is installed at the more eastern door of the hallway.
Prereferences	: akref BD_14
Postreferences	: msref HallwaySection : M_HS3 dkref HallwaySection : D_HS1 msref HallwaySection : M_HS4

Object *westMd* : *MotionDetector*

NL	: motion detectors
Intention	: This object represents the motion detector in a hallway section which is installed at the more western door of the hallway.
Prereferences	: akref BD_14
Postreferences	: msref HallwaySection : M_HS3 dkref HallwaySection : D_HS1 msref HallwaySection : M_HS4

Object *middleMd* : *MotionDetector*

NL	: motion detectors
Intention	: This object represents the motion detector in a hallway section which is installed in the middle of the hallway.
Prereferences	: akref BD_14
Postreferences	: msref HallwaySection : M_HS3 dkref HallwaySection : D_HS1 msref HallwaySection : M_HS4

Object *eastDoor* : *Door*

Intention	: This object represents the door at the eastern end of a hallway section.
Prereferences	: akref BD_13
Postreferences	: dkref Floor : D_F7 dkref Floor : D_F8 msref HallwaySection : M_HS4

Object *westDoor* : *Door*

Intention	: This object represents the door at the western end of a hallway section.
Prereferences	: akref BD_13
Postreferences	: dkref Floor : D_F7 dkref Floor : D_F8 msref HallwaySection : M_HS4

Time Constant *T_HS1*

NL	: near-to-light delay
Intention	: This constant represents the maximal delay between a time point a person is near a hallway door indicating that he wants to enter a hallway section and the time point the ceiling light group of this hallway section is on.
Scope	: ev
Prereferences	: pdref U14
Postreferences	: msref HallwaySection : M_HS1 rsref HallwaySection : R_HS1

Time Constant *T_HS2*

NL	: nocontrol-to-light delay
Intention	: This constant represents the maximal delay between a time point the ceiling light group of a hallway section is neither automatically nor manually controllable and the time point the ceiling light group in this hallway section is on.
Scope	: ev
Prereferences	: pdref NF5
Postreferences	: rsref HallwaySection : R_HS2

Time Constant *T_OccReact*

Intention	: This time constant represents the maximal time delay between the time point a hallway section becomes occupied and the time point one of the three motion detectors installed in a hallway section delivers a 1.
Scope	: ev
Postreferences	: dkref HallwaySection : D_HS1 msref HallwaySection : M_HS5

Timed Predicate *nearHallwayDoor*

NL	: near a hallway door
Intention	: This predicate is true if and only if a person is near a hallway door to a hallway section indicating that he wants to enter this hallway section.
Scope	: eh
Prereferences	: pdref U14
Postreferences	: rsref HallwaySection : R_HS1

Timed Predicate *hwLightUncontroll*

NL	: hallway light uncontrollable
Intention	: This predicate is true if and only if the ceiling light group in a hallway section is neither automatically nor manually controllable.
Scope	: eh
Prereferences	: pdref NF5
Postreferences	: rsref HallwaySection : R_HS2

Domain Knowledge

Property *D_HS1*

Formal : $\Box(occFM.occEnv \Leftrightarrow_{\leq T\_OccReact} (eastMd.usedValue = 1 \vee westMd.usedValue = 1 \vee middleMd.usedValue = 1))$

NL : Whenever a hallway section is occupied then within T_OccReact milliseconds the motion detector installed at the more eastern door or the motion detector installed at the more western door or the motion detector installed in the middle of the hallway delivers a 1 and one of these motion detectors delivers a 1 at least as long as the hallway section is occupied.
And whenever a hallway section is not occupied then within T_OccReact milliseconds all three motion detectors installed in a hallway section delivers a 0 and remain delivering a 0 at least as long as the hallway section is not occupied.

Propreferences : rsref Area : R_A2

Requirement Specifications

Property *R_HS1*

Formal : $\Box(nearHallwayDoor \Rightarrow_{\leq T\_HS1} light[1].sl.noMalSens.envEntity = 1)$

NL : Whenever a person is continuously near to a hallway door of a hallway section for at least *T_HS_1* time units indicating that he wants to enter the hallway section, then the ceiling light group in this hallway section will be on within this time span, and remains on at least as long as the preconditions are fulfilled.

Property *R_HS2*

Formal : $\Box(hwLightUncontroll \Rightarrow_{\leq T\_HS2} light[1].sl.noMalSens.envEntity = 1)$

NL : Whenever the ceiling light group in a hallway section is controllable neither automatically nor manually for at least *T_HS_2* time units, then eventually within this time span, the ceiling light group in this hallway section is on and remains so at least as long as the ceiling light group is controllable neither 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 ceiling ight group in a hallway section 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 : $\Box(occFM.noMalOcc.areaEmpty \leftrightarrow$
$((eastMd.usedValue = 0) \wedge$
$(westMd.usedValue = 0) \wedge$
$(middleMd.usedValue = 0)))$

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

Propreferences : rsref Area : R_A2

Property *M_HS4*

Formal : $\Box(CPFM.indicatorMalFct.informCondMach \leftrightarrow$
$eastMd.malSens.malDetMach \vee$
$westMd.malSens.malDetMach \vee$
$middleMd.malSens.malDetMach \vee$
$eastDoor.doorClosedContact.malSens.malDetMach \vee$
$westDoor.doorClosedContact.malSens.malDetMach \vee$
$light[1].sl.malSens.malDetMach \vee$
$light[1].pulse.malAct.malDetMach \vee$
$light[1].csa.malAct.malDetMach)$

NL : The facility manager should be informed about a malfunction if one of the following sensors or actuators has a malfunction:

- the *motion detector* installed at the more eastern door of a hallway section
- the *motion detector* installed at the more western door of a hallway section
- the *motion detector* installed in the middle of a hallway section
- the *door closed contact* of the door at the eastern end of a hallway section
- the *door closed contact* of the door at the western end of a hallway section
- the *status line* of the ceiling light group
- the *pulse* of the ceiling light group
- the *control system active* of the ceiling light group

Property M_HS5

Formal	: $\Box(T\_A1 \geq T\_OccReact + T\_A2 + light[1].pulse.noMalAct.reactionTime)$
NL	: The time constant T_A1 is always greater than the sum of the time constants T_A2, T_OccReact, and the reaction time of the pulse of a ceiling light group.
Propreferences	: rsref Area : R_A2

Template Declaration Room

Intention

This class provides all specifications concerning the control of ceiling light groups common to any kind of room. In more detail this means that the following cases are considered

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

In each room there are two dimmable ceiling light groups that are controlled by the machine:

- a dimmable ceiling light group at the window and
- a dimmable ceiling light group at the wall to the hallway.

Furthermore, in each room there is a room 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.

Postreferences :	lsref Office : Room
	lsref NoOffice : Room
	lsref Floor : ROOMS

Formal Parameters

Function *numDoorHallway* $\rightarrow$ *NAT*

Intention	: This parameter represents the number of doors of a room to a hallway section.
Default Value	: 1
Postreferences	: lsref Room : hallwayDoor
	msref Room : M_R13
	msref Room : M_R14
	lsref Office : Room
	lsref NoOffice : Room
	lsref Floor : ROOMS

Base Classes

Class *Area*(*numLights* = 2, *numPB* = 1)

Intention : All definitions of description class *Area* are inherited. There are two ceiling light groups in a room that are controlled by the machine:

- *light*[1] represents the ceiling light group near the window and
- *light*[2] represents the ceiling light group near the wall to a hallway section.

Signature

Object *Area* :: *light* : *DimmableCeilingLightGroup*

NL : window light
Intention : The ceiling light groups in a room are *dimmable* ceiling light groups.
Postreferences : msref Room : M_R8
msref Room : M_R8b
msref Room : M_R9
msref Room : M_R10
msref Room : M_R13
msref Room : M_R14

Object *CPRoom* : *RoomCtrlPanel*

NL : light control panel room
Intention : This object represents the room control panel.
Prereferences : pdref U5
pdref U6
pdref U7
pdref U9
Postreferences : msref Room : M_R3
msref Room : M_R7
msref Room : M_R5
msref Room : M_R2
msref Room : M_R6
msref Room : M_R4
msref Room : M_R8
msref Room : M_R8b
msref Room : M_R9
msref Room : M_R13

Object *md* : *MotionDetector*

NL	: motion detector
Intention	: This object represents the room motion detector.
Prereferences	: akref BD_8
	akref BD_9
	akref BD_10
	akref BD_11
Postreferences	: msref Room : M_R11
	msref Room : M_R12
	msref Room : M_R13
	msref Room : M_R14

Object *hallwayDoor* : *Door*[*numDoorHallway*]

Intention	: Each object represents one door of a room to a hallway section.
Prereferences	: akref BD_7
	akref BD_9
	akref BD_10
	akref BD_11
Postreferences	: msref Room : M_R13
	msref Room : M_R14

Object *occRoom* : *Occupancy*

Intention	: This object represents the part of the light control concerning occupancy in a room as it is defined concerning the handling of light scenes. Note that in the description class *Area* another version of the aspect occupancy is specified representing the view of the facility manager w.r.t. the aspect occupancy.
Prereferences	: pdref U1
	pdref U2
	pdref U3
	pdref U4
Postreferences	: msref Room : M_R5
	msref Room : M_R6
	msref Room : M_R7
	msref Room : M_R4
	msref Room : M_R12
	msref Room : M_R8
	msref Room : M_R8b
	msref Room : M_R9

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
Prereferences	: pdref U2
Postreferences	: msref Room : I_M_R2 msref Room : M_R7 msref Room : M_R5 msref Room : M_R6 msref Room : M_R8 msref Room : M_R8b

Timed Function *curDefaultLightScene* → *LightScene*

NL	: current standard light scene
Intention	: This function represents the current default light scene that is established by the machine if light scenes are enabled.
Scope	: mh
Prereferences	: pdref U4 pdref NF2
Postreferences	: msref Room : I_M_R1 msref Room : I_M_R2 msref Floor : M_F2 msref Room : M_R6

Timed Function *malDefaultLightScene* → *LightScene*

NL	: fault standard light scene
Intention	: This function represents the light scene that is the default light scene in the case of a malfunction of the outdoor light sensors.
Scope	: mh
Prereferences	: pdref NF2
Postreferences	: msref Floor : M_F2 msref Room : M_R1

Timed Function *noMalDefaultLightScene* → *LightScene*

NL	: no fault standard light scene
Intention	: This function represents the light scene that is the default light scene in the case that the outdoor light sensors have no malfunctions.
Scope	: mh
Postreferences	: msref Room : I_M_R1 msref Room : M_R2

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

Intention	: This function determines the light scene which is identified by the given name.
Scope	: mh
Postreferences	: msref Room : M_R7 msref Room : M_R5

Timed Predicate *OLSMalFct*

Intention	: This predicate is true if and only if an outdoor light sensor that is used by a room has a malfunction. Which outdoor light sensors are used by a room is specified in the description class *Floor*.
Scope	: mh
Prereferences	: pdref U8 pdref NF2
Postreferences	: msref Floor : M_F13 msref Floor : M_F2 msref Floor : M_F14 msref Floor : M_F15 msref Floor : M_F16 msref Floor : M_F17 msref Floor : M_F18 msref Floor : M_F19 msref Floor : M_F20 msref Floor : M_F21 msref Floor : M_F22 msref Floor : M_F23 msref Floor : M_F24 msref Floor : M_F25 msref Floor : M_F6 msref Floor : M_F7 msref Floor : M_F8 msref Floor : M_F9 msref Floor : M_F10 msref Floor : M_F11 msref Floor : M_F12 msref Room : M_R13 msref Room : M_R14

Timed Predicate *closedDoorContactNeighborMal*

Intention	: This predicate is true if and only if a door closed contact of a door to a neighbor room has a malfunction. This predicate is refined in the description classes *OfficeNoNeighbor*, *OfficeNeighbor*, *NoOfficeNoNeighbor*, *NoOfficeNeighbor*.
Scope	: mh
Postreferences	: msref OfficeNeighbor : M_ON1 msref NoOfficeNeighbor : M_NON1 msref OfficeWithoutNeighbor : M_ONN1 msref NoOfficeWithoutNeighbor : M_NONN1 msref Room : M_R13 msref Room : M_R14

Function $evalLightScene : LightScene, LightScene :: CEIL\_LIGHT\_GROUPS, LIGHT\_VALUES \to PERCENT$

NL	: the dimmer setting of $2 w.r.t. the light scene 1$
Intention	: This function represents the computation of the settings for a given ceiling light group (represented by the second argument) depending on a specific light scene (represented by the first argument) and the current day light the machine assumes to be available in a room(represented by the third argument). The value of this function represents the dimmer settings and additionally the corresponding ceiling light groups has to be on if and only if the dimmer setting is greater than 0.
Scope	: mh
Prereferences	: pdref U2 pdref FM1
Postreferences	: msref Room : M_R8 msref Room : M_R8b

Machine Specifications

Property *I_M_R1*

Formal	: $curDefaultLightScene = noMalDefaultLightScene$
NL	: Initially the current default light scene is the default light scene for the case that the outdoor light sensors have no malfunction.

Property *I_M_R2*

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

Property *M_R1*

Formal : $\Box(malDefaultLightScene.name =$ "MAL_DEFAULT"$\wedge$
$malDefaultLightScene.ambLightLevel = 14 \wedge$
$malDefaultLightScene.option = LightScene :: BOTH)$

NL : The default light scene in the case of a malfunction of the outdoor light sensors is specified as follows:

- the *name* is *MAL_DEFAULT*
- the desired *ambient light level* is 14 lux
- the *option* is *BOTH*

Prereferences : pdref U1

Property *M_R2*

Formal : $\Box(noMalDefaultLightScene = CPRoom.defaultLightScene.checked)$

NL : The default light scene in the case that the outdoor light sensors used by a room have no malfunction always corresponds with the default light scene determined by a user using the room control panel.

Prereferences : pdref U6

Property *M_R3*

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

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

Prereferences : pdref U1

Property *M_R4*

Formal : $\Box(occRoom.noMalOcc.delay = CPRoom.reoccDelay.checked)$

NL : The time delay used to determine occupancy as it seen by a user corresponds always with the value entered by a user for the time within the reoccupation of a room leads to a reestablishing of the previous light scene.

Prereferences : pdref U7
pdref U9

Property M_R5

Formal	: $\Box(([\neg occRoom.noMalOcc.areaEmpty] \wedge \blacklozenge_{\leq CPRoom.reoccDelay.checked} \neg occRoom.noMalOcc.areaEmpty) \rightarrow curLightScene = determineLightScene(CPRoom.chosenLightScene.checked))$
NL	: Whenever the room becomes non-empty now and has been non-empty 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.
Prereferences	: pdref U3

Property M_R6

Formal	: $\Box(([\neg occRoom.noMalOcc.areaEmpty] \wedge \blacksquare_{> CPRoom.reoccDelay.checked} occRoom.noMalOcc.areaEmpty) \rightarrow curLightScene = curDefaultLightScene)$
NL	: Whenever the room becomes non-empty now and has been continuously empty for the last T_RL_2 milliseconds then the current light scene is the current default light scene.
Prereferences	: pdref U4

Property M_R7

Formal	: $\Box(\forall l \in LightScene : ((occRoom.occUsed \wedge \neg[occRoom.occUsed] \wedge [determineLightScene(CPRoom.chosenLightScene.checked) = l]) \rightarrow curLightScene = l))$
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.
Prereferences	: pdref U2 pdref U10

Property *M_R8*

Formal : $\Box((occRoom.occUsed \wedge \neg manualControl \wedge CPRoom.lightSceneSwitch.checked = 1)$
$\rightarrow ((light[1].sl.usedValue = 0 \wedge evalLightScene(curLightScene, LightScene :: WINDOW, dayLight) \neq 0)$
$\rightarrow [light[1].pulse.noMalAct.adjustedEntity = 1]) \wedge$
$((light[1].sl.usedValue = 1 \wedge evalLightScene(curLightScene, LightScene :: WINDOW, dayLight) = 0)$
$\rightarrow [light[1].pulse.noMalAct.adjustedEntity = 1]) \wedge$
$(light[1].dimmer.noMalAct.adjustedEntity = evalLightScene(curLightScene, LightScene :: WINDOW, dayLight)))$

NL : Whenever the room is occupied and the light is not controlled manually and light scenes are enabled by the control panel, the window ceiling light group is dimmed and switched on or off as necessary to achieve the ambient light level of the current light scene.
This means in the long form:

- if the window ceiling light group should be on, i.e., the computed dimmer value is not equal to 0, and the window ceiling light group is off, a pulse is given for the window ceiling light group
- if the window ceiling light group should be off, i.e., the computed dimmer value is equal to 0, and the window ceiling light group is on, a pulse is given for the window ceiling light group
- the dimmer value of the window ceiling light group is set to the computed one

Property M_R8b

Formal : $\Box((occRoom.occUsed \wedge \neg manualControl \wedge$
$CPRoom.lightSceneSwitch.checked = 1)$
$\rightarrow ((light[2].sl.usedValue = 0 \wedge$
$evalLightScene(curLightScene,$
$LightScene :: WALL,$
$dayLight) \neq 0)$
$\rightarrow [light[2].pulse.noMalAct.adjustedEntity = 1]) \wedge$
$((light[2].sl.usedValue = 1 \wedge$
$evalLightScene(curLightScene,$
$LightScene :: WALL,$
$dayLight) = 0)$
$\rightarrow [light[2].pulse.noMalAct.adjustedEntity = 1]) \wedge$
$(light[2].dimmer.noMalAct.adjustedEntity =$
$evalLightScene(curLightScene,$
$LightScene :: WALL,$
$dayLight)))$

NL : Whenever the room is occupied and the light is not controlled manually and light scenes are enabled by the control panel, the wall ceiling light group is dimmed and switched on or off as necessary to achieve the ambient light level of the current light scene.
This means in the long form:

- if the wall ceiling light group should be on, i.e., the computed dimmer value is not equal to 0, and the wall ceiling light group is off, a pulse is given for the wall ceiling light group
- if the wall ceiling light group should be off, i.e., the computed dimmer value is equal to 0, and the wall ceiling light group is on, a pulse is given for the wall ceiling light group
- the dimmer value of the wall ceiling light group is set to the computed one

Property *M_R9*

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

NL : Whenever the room is occupied and the light is not manually controlled and light scenes are disabled using the the control panel, the window and wall lights are completely switched off or on as selected by a user using the control panels ceiling light switches.
This means in the long form:

- if the window ceiling light group should be on:
  - its dimmer is set to 100 percent
  - if the window ceiling light group is off a pulse is given
- if the window ceiling light group should be off:
  - its dimmer is set to 0 percent
  - if the window ceiling light group is on a pulse is given
- if the wall ceiling light group should be on:
  - its dimmer is set to 100 percent
  - if the wall ceiling light group is off a pulse is given
- if the wall ceiling light group should be off:
  - its dimmer is set to 0 percent
  - if the wall ceiling light group is on a pulse is given

In short: Under the mentioned conditions the light is controlled by the ceiling light switches.

Property *M_R10*

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

NL : As long as the light is controlled manually, all settings of the window and wall ceiling light group remain unchanged. This means:

- no pulse is given for the window ceiling light group to switch it on or off
- the dimmer setting of the window ceiling light group is not changed
- no pulse is given for the wall ceiling light group to switch it on or off
- the dimmer setting of the wall ceiling light group is not changed

Property *M_R11*

Formal : $\Box(occFM.noMalOcc.areaEmpty \leftrightarrow md.usedValue = 0)$

NL : The machine always assumes that there is currently no person in a room concerning the specification of occupancy as it is seen by the facility manager if and only if the value of the motion detector is 0.

Property *M_R12*

Formal : $\Box(occRoom.noMalOcc.areaEmpty \leftrightarrow md.usedValue = 0)$

NL : The machine always assumes that there is currently no person in a room concerning the specification of occupancy as it is seen by the user if and only if the value of the motion detector is 0.

Property *M_R13*

Formal : $\Box(CPRoom.indicatorMalFct.informCondMach \leftrightarrow$
$md.malSens.malDetMach \vee$
$light[1].sl.malSens.malDetMach \vee$
$light[1].pulse.malAct.malDetMach \vee$
$light[1].csa.malAct.malDetMach \vee$
$light[1].dimmer.malAct.malDetMach \vee$
$light[2].sl.malSens.malDetMach \vee$
$light[2].pulse.malAct.malDetMach \vee$
$light[2].csa.malAct.malDetMach \vee$
$light[2].dimmer.malAct.malDetMach \vee$
$md.malSens.malDetMach \vee$
$\exists n \in NAT(numDoorHallway):$
$hallwayDoor[n].doorClosedContact.malSens.malDetMach \vee$
$closedDoorContactNeighborMal \vee$
$OLSMalFct)$

NL : A user should be informed about a malfunction if one of the following sensors or actuators has a malfunction:

- the *status line* of the window ceiling light group
- the *pulse* of the window ceiling light group
- the *control system active* of the window ceiling light group
- the *dimmer* of the window ceiling light group
- the *status line* of the wall ceiling light group
- the *pulse* of the wall ceiling light group
- the *control system active* of the wall ceiling light group
- the *dimmer* of the wall ceiling light group
- the *motion detector*
- the *door closed contact* of a door to a hallway section
- the *door closed contact* of a door to a neighbor room
- the *outdoor light sensors* used by a room

Property *M_R14*

Formal : $\Box(CPFM.indicatorMalFct.informCondMach \leftrightarrow$
$md.malSens.malDetMach \vee$
$light[1].sl.malSens.malDetMach \vee$
$light[1].pulse.malAct.malDetMach \vee$
$light[1].csa.malAct.malDetMach \vee$
$light[1].dimmer.malAct.malDetMach \vee$
$light[2].sl.malSens.malDetMach \vee$
$light[2].pulse.malAct.malDetMach \vee$
$light[2].csa.malAct.malDetMach \vee$
$light[2].dimmer.malAct.malDetMach \vee$
$md.malSens.malDetMach \vee$
$\exists n \in NAT(numDoorHallway):$
$hallwayDoor[n].doorClosedContact.malSens.malDetMach \vee$
$closedDoorContactNeighborMal \vee$
$OLSMalFct)$

NL : The facility manager should be informed about a malfunction if one of the following sensors or actuators has a malfunction:

- the *status line* of the window ceiling light group
- the *pulse* of the window ceiling light group
- the *control system active* of the window ceiling light group
- the *dimmer* of the window ceiling light group
- the *status line* of the wall ceiling light group
- the *pulse* of the wall ceiling light group
- the *control system active* of the wall ceiling light group
- the *dimmer* of the wall ceiling light group
- the *motion detector*
- the *door closed contact* of a door to a hallway section
- the *door closed contact* of a door to a neighbor room
- the *outdoor light sensors* used by a room

Template Declaration NeighborRoomConnections

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 neighbor room. Such rooms are equipped additionally with

- *numDoorNeighbor* many doors to neighbor 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 an office or not.

Postreferences	: lsref OfficeNeighbor : NeighborRoomConnections
	lsref NoOfficeNeighbor : NeighborRoomConnections

Formal Parameters

Function $numDoorNeighbor \rightarrow NAT$

Intention	: This parameter represents the number of doors of a room to neighbor rooms.
Postreferences	: lsref NeighborRoomConnections : neighborDoor
	dkref NeighborRoomConnections : D_NRC1

Signature

Object $neighborDoor : Door[numDoorNeighbor]$

Intention	: Each object represents one door of a room to a neighbor room.
Prereferences	: akref BD_7
Postreferences	: dkref Floor : D_F1
	dkref Floor : D_F2
	dkref Floor : D_F3
	dkref Floor : D_F4
	dkref Floor : D_F5
	dkref Floor : D_F6
	msref OfficeNeighbor : M_ON1
	msref NoOfficeNeighbor : M_NON1

Domain Knowledge

Property *D_NRC1*

Formal	: $\Box(numDoorNeighbor \leq 2)$
NL	: There are always at most two doors leading to a neighbor room.
Prereferences	: akref BD_7

Template Declaration 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.

Postreferences	: lsref OfficeNeighbor : Office
	lsref OfficeWithoutNeighbor : Office

Formal Parameters

Function *numDoorHallway* $\rightarrow$ *NAT*

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

Base Classes

Class *Room*

Intention	: All definitions of description class *Room* are inherited.

Requirement Specifications

Property *R_O1*

NonFormal	: The control panel should be movable in the offices like a telephone with a cord.
Prereferences	: pdref U11

Template Declaration OfficeNeighbor

Intention

This class provides all definitions specific to all offices which are connected by at least one door to a neighbor room. Therefore, in addition to the definitions of the description class *Office* the definitions of the description class *NeighborRoomConnections* are inherited.

Postreferences : lsref Floor : o435
lsref Floor : o433
lsref Floor : o431
lsref Floor : o425
lsref Floor : o423
lsref Floor : o421
lsref Floor : o416
lsref Floor : o414

Formal Parameters

Function $numDoorHallway \rightarrow NAT$

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

Postreferences : lsref OfficeNeighbor : Office

Function $numDoorNeighbor \rightarrow NAT$

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

Postreferences : lsref OfficeNeighbor : NeighborRoomConnections
msref OfficeNeighbor : M_ON1

Base Classes

Class *Office*($numDoorHallway = numDoorHallway$)

Intention : All definitions of description class *Office* are inherited.

Class *NeighborRoomConnections*($numDoorNeighbor = numDoorNeighbor$)

Intention : All definitions of description class *NeighborRoomConnections* are inherited.

Machine Specifications

Property *M_ON1*

Formal : $\Box(closedDoorContactNeighborMal \leftrightarrow$
$\exists n \in NAT(numDoorNeighbor) :$
$neighborDoor[n].doorClosedContact.malSens.malDetMach)$

NL : A door closed contact of a door to a neighbor room has a malfunction if and only if at least one of these contacts has a malfunction.

Template Declaration OfficeWithoutNeighbor

Intention

This class provides all definitions specific to rooms that are offices and which are not connected by a door to a neighbor room.

Postreferences : lsref Floor : o429
lsref Floor : o419
lsref Floor : o417
lsref Floor : o415
lsref Floor : o424
lsref Floor : o412

Formal Parameters

Function $numDoorHallway \rightarrow NAT$

Intention : This parameter represents the number of doors of a room to a hallway section.
Postreferences : lsref OfficeWithoutNeighbor : Office

Base Classes

Class *Office*($numDoorHallway = numDoorHallway$)

Intention : All definitions of description class *Office* are inherited.

Machine Specifications

Property *M_ONN1*

Formal : $\Box(closedDoorContactNeighborMal \leftrightarrow FALSE)$
NL : A door closed contact of a door to a neighbor room has never a malfunction.
Note that an object of this description class has no door to a neighbor room.

Template Declaration 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.

Postreferences	: lsref NoOfficeNeighbor : NoOffice
	lsref NoOfficeWithoutNeighbor : NoOffice

Formal Parameters

Function $numDoorHallway \to NAT$

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

Base Classes

Class *Room*

Intention	: All definitions of description class *Room* are inherited.

Requirement Specifications

Property *R_NO1*

NonFormal	: The control panel should be installed near a door to a hallway section.
Prereferences	: pdref U12

Template Declaration NoOfficeNeighbor

Intention

This class provides all definitions specific to rooms that are no offices and which are connected by at least one door to a neighbor room. In addition to the definitions of the description class *NoOffice* the definitions of the description class *NeighborRoomConnections* are inherited.

Postreferences : lsref Floor : cl422
lsref Floor : cl418

Formal Parameters

Function $numDoorHallway \rightarrow NAT$

Intention : This parameter represents the number of doors of a room to the hallway.
Postreferences : lsref NoOfficeNeighbor : NoOffice

Function $numDoorNeighbor \rightarrow NAT$

Intention : This parameter represents the number of doors of a room to neighbor rooms.
Postreferences : lsref NoOfficeNeighbor : NeighborRoomConnections
msref NoOfficeNeighbor : M_NON1

Base Classes

Class $NoOffice(\ numDoorHallway = numDoorHallway\ )$

Intention : All definitions of a description class *NoOffice* are inherited.

Class $NeighborRoomConnections(\ numDoorNeighbor = numDoorNeighbor\ )$

Intention : All definitions of description class *NeighborRoomConnections* are inherited.

Machine Specifications

Property *M_NON1*

Formal : $\Box(closedDoorContactNeighborMal \leftrightarrow \exists n \in NAT(numDoorNeighbor) : neighborDoor[n].doorClosedContact.malSens.malDetMach)$
NL : A door closed contact of a door to a neighbor room has a malfunction if and only if at least one of these contacts has a malfunction.

Template Declaration NoOfficeWithoutNeighbor

Intention

This class provides all definitions specific to rooms that are no offices and which are not connected by a door to a neighbor room.

Postreferences : lsref Floor : m427
lsref Floor : hl411
lsref Floor : cl426
lsref Floor : hl410

Formal Parameters

Function $numDoorHallway \rightarrow NAT$

Intention : This parameter represents the number of doors of a room to a hallway section.
Postreferences : lsref NoOfficeWithoutNeighbor : NoOffice

Base Classes

Class *NoOffice*($numDoorHallway = numDoorHallway$)

Intention : All definitions of description class *Office* are inherited.

Machine Specifications

Property *M_NONN1*

Formal : $\Box(closedDoorContactNeighborMal \leftrightarrow FALSE)$
NL : A door closed contact of a door to a neighbor room has never a malfunction.
Note that an object of this description class has no door to a neighbor room.

Template Declaration 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.

Postreferences : lsref Floor : scw
lsref Floor : sce

Signature

Object *md* : *MotionDetector*

Intention : This object represents a motion detector placed in a staircase above the door of the landing that leads to the adjacent hallway section.
Prereferences : akref BD_16
Postreferences : msref Floor : M_F3
msref Floor : M_F4

Template Declaration 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 *o435* : *OfficeNeighbor*(*numDoorHallway* = 1, *numDoorNeighbor* = 1)

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

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

Prereferences : akref Figure1
Postreferences : dkref Floor : D_F1
lsref Floor : ROOMS
msref Floor : M_F6

Object *o433* : *OfficeNeighbor*(*numDoorHallway* = 1, *numDoorNeighbor* = 2)

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

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

Prereferences : akref Figure1
Postreferences : dkref Floor : D_F1
dkref Floor : D_F2
lsref Floor : ROOMS
msref Floor : M_F7

Object *o431* : *OfficeNeighbor*(*numDoorHallway* = 1, *numDoorNeighbor* = 1)

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

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

Prereferences : akref Figure1
Postreferences : dkref Floor : D_F2
lsref Floor : ROOMS
msref Floor : M_F8

Object *o429* : *OfficeWithoutNeighbor*(*numDoorHallway* = 1)

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

- one door to the hallway

Prereferences : akref Figure1
Postreferences : lsref Floor : ROOMS
msref Floor : M_F9

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

Intention : This object represents the meeting room with room number 427, which is equipped with

- one door to the hallway

Prereferences : akref Figure1
Postreferences : lsref Floor : ROOMS
msref Floor : M_F10

Object *o425* : *OfficeNeighbor*(*numDoorHallway* = 1, *numDoorNeighbor* = 1)

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

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

Prereferences : akref Figure1
Postreferences : dkref Floor : D_F3
lsref Floor : ROOMS
msref Floor : M_F11

Object *o423* : *OfficeNeighbor*(*numDoorHallway* = 1, *numDoorNeighbor* = 2)

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

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

Prereferences : akref Figure1
Postreferences : dkref Floor : D_F3
dkref Floor : D_F4
lsref Floor : ROOMS
msref Floor : M_F12

Object *o421* : *OfficeNeighbor*(*numDoorHallway* = 1, *numDoorNeighbor* = 1)

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

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

Prereferences : akref Figure1
Postreferences : dkref Floor : D_F4
lsref Floor : ROOMS
msref Floor : M_F13

Object *o419* : *OfficeWithoutNeighbor*(*numDoorHallway* = 1)

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

- one door to the hallway

Prereferences : akref Figure1
Postreferences : lsref Floor : ROOMS
msref Floor : M_F14

Object *o417* : *OfficeWithoutNeighbor*(*numDoorHallway* = 1)

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

- one door to the hallway

Prereferences : akref Figure1
Postreferences : lsref Floor : ROOMS
msref Floor : M_F15

Object *o415* : *OfficeWithoutNeighbor*(*numDoorHallway* = 1)

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

- one door to the hallway

Prereferences : akref Figure1
Postreferences : lsref Floor : ROOMS
msref Floor : M_F16

Object *hl411* : *NoOfficeWithoutNeighbor*(*numDoorHallway* = 2)

Intention : This object represents the hardware lab with room number 411, which is equipped with

- two doors to the hallway

Prereferences : akref Figure1
Postreferences : lsref Floor : ROOMS
msref Floor : M_F17

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

Intention : This object represents the computer lab with room number 426, which is equipped with

- one door to the hallway

Prereferences : akref Figure1
Postreferences : lsref Floor : ROOMS
msref Floor : M_F18

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

Intention : This object represents the office with room number 424, which is equipped with

- one door to the hallway

Prereferences : akref Figure1
Postreferences : lsref Floor : ROOMS
msref Floor : M_F19

Object *cl422* : *NoOfficeNeighbor*(*numDoorHallway* = 1, *numDoorNeighbor* = 1)

Intention : This object represents the computer lab with room number 424, which is equipped with

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

Prereferences : akref Figure1
Postreferences : lsref Floor : ROOMS
msref Floor : M_F20

Object *cl418* : *NoOfficeNeighbor*(*numDoorHallway* = 1, *numDoorNeighbor* = 2)

Intention : This object represents the computer lab with room number 418, which is equipped with

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

Prereferences : akref Figure1
Postreferences : dkref Floor : D_F5
lsref Floor : ROOMS
msref Floor : M_F21

Object *o416* : *OfficeNeighbor*(*numDoorHallway* = 1, *numDoorNeighbor* = 2)

Intention : This object represents the office with room number 416, which is equipped with

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

Prereferences : akref Figure1
Postreferences : dkref Floor : D_F5
dkref Floor : D_F6
lsref Floor : ROOMS
msref Floor : M_F22

Object *o414* : *OfficeNeighbor*(*numDoorHallway* = 1, *numDoorNeighbor* = 1)

Intention : This object represents the office with room number 414, which is equipped with

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

Prereferences : akref Figure1
Postreferences : dkref Floor : D_F6
lsref Floor : ROOMS
msref Floor : M_F23

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

Intention : This object represents the office with room number 412, which is equipped with

- one door to the hallway

Prereferences : akref Figure1
Postreferences : lsref Floor : ROOMS
msref Floor : M_F24

Object *hl410* : *NoOfficeWithoutNeighbor*(*numDoorHallway* = 3)

Intention : This object represents the hardware lab with room number 410, which is equipped with

- three doors to the hallway

Prereferences : akref Figure1
Postreferences : lsref Floor : ROOMS
msref Floor : M_F25

Object *h1* : *HallwaySection*(*numPB* = 3)

NL : hallway 1
Intention : This object represents the hallway in the east of floor 32/4. There are three push buttons for controlling the ceiling light group in this hallway section.
Prereferences : akref Figure1
akref BD_6
Postreferences : lsref Floor : HALLWAYS
dkref Floor : D_F7

Object *h2* : *HallwaySection*(*numPB* = 3)

NL	: hallway 2
Intention	: This object represents the hallway in the middle of floor 32/4. There are three push buttons for controlling the ceiling light group in this hallway section.
Prereferences	: akref Figure1 akref BD_6
Postreferences	: lsref Floor : HALLWAYS dkref Floor : D_F7 dkref Floor : D_F8

Object *h3* : *HallwaySection*(*numPB* = 3)

NL	: hallway 3
Intention	: This object represents the hallway in the west of floor 32/4. There are three push buttons for controlling the ceiling light group in this hallway section.
Prereferences	: akref Figure1 akref BD_6
Postreferences	: lsref Floor : HALLWAYS dkref Floor : D_F8

Object *ols1* : *OutdoorLightSensor*

NL	: outdoor light sensor 1
Intention	: This object represents the outdoor light sensor in the north-east direction.
Prereferences	: akref Figure1 akref BD_6
Postreferences	: msref Floor : M_F1 msref Floor : M_F6 msref Floor : M_F7 msref Floor : M_F8 msref Floor : M_F9 msref Floor : M_F10

Object *ols2* : *OutdoorLightSensor*

NL	: outdoor light sensor 2
Intention	: This object represents the outdoor light sensor in the south-west direction.
Prereferences	: akref Figure1 akref BD_6
Postreferences	: msref Floor : M_F1 msref Floor : M_F18 msref Floor : M_F19 msref Floor : M_F20

Object *ols3* : *OutdoorLightSensor*

NL	: outdoor light sensor 3
Intention	: This object represents the outdoor light sensor in the north-west direction.
Prereferences	: akref Figure1 akref BD_6
Postreferences	: msref Floor : M_F1 msref Floor : M_F20 msref Floor : M_F13 msref Floor : M_F14 msref Floor : M_F15 msref Floor : M_F16

Object *ols4* : *OutdoorLightSensor*

NL	: outdoor light sensor 4
Intention	: This object represents the outdoor light sensor in the south-east direction.
Prereferences	: akref Figure1 akref BD_6
Postreferences	: msref Floor : M_F1 msref Floor : M_F21 msref Floor : M_F22 msref Floor : M_F23 msref Floor : M_F24

Object *ols5* : *OutdoorLightSensor*

NL	: outdoor light sensor 5
Intention	: This object represents the outdoor light sensor in the north direction.
Prereferences	: akref Figure1 akref BD_6
Postreferences	: msref Floor : M_F1 msref Floor : M_F17 msref Floor : M_F10 msref Floor : M_F11 msref Floor : M_F12 msref Floor : M_F13

Object *ols6* : *OutdoorLightSensor*

NL	: outdoor light sensor 6
Intention	: This object represents the outdoor light sensor in the south direction.
Prereferences	: akref Figure1 akref BD_6
Postreferences	: msref Floor : M_F1 msref Floor : M_F24 msref Floor : M_F25

Object *scw* : *Staircase*

NL	: western staircase
Intention	: This object represents the staircase in the west of floor 32/4.
Prereferences	: akref Figure1 akref BD_6
Postreferences	: msref Floor : M_F4

Object *sce* : *Staircase*

NL	: eastern staircase
Intention	: This object represents the staircase in the east of floor 32/4.
Prereferences	: akref Figure1 akref BD_6
Postreferences	: msref Floor : M_F3

Object *indicatorMalFctWestStaircase* : *InformSensActMalFct*

Intention	: This object represents the handling of a malfunction of the motion detector in the western staircase.
Postreferences	: msref Floor : M_F4

Object *indicatorMalFctEastStaircase* : *InformSensActMalFct*

Intention	: This object represents the handling of a malfunction of the motion detector in the eastern staircase.
Postreferences	: msref Floor : M_F3

Domain $ROOMS = \{o435, o433, o431, o429, o425, o423, o421, o419, o417, o415, o424, o416, o414, o412, m427, hl411, cl426, cl422, cl418, hl410\} \subseteq Room$

Intention	: This domain contains all rooms of the specified floor.
Prereferences	: akref Figure1 akref BD_6
Postreferences	: msref Floor : M_F2 msref Floor : M_F5

Domain $HALLWAYS = \{h1, h2, h3\} \subseteq HallwaySection(\ numPB = 3\ )$

Intention	: This domain contains all the hallway sections that have to be considered.
Prereferences	: akref Figure1 akref BD_6
Postreferences	: rsref Floor : R_F1 msref Floor : M_F5

Time Constant *TmalReact*

Intention	: The time delay between a malfunction of the outdoor light sensor and switching on the lights if a hallway section is occupied.
Scope	: ev
Prereferences	: pdref NF3
Postreferences	: rsref Floor : R_F1

Timed Predicate *outLightSensorFaultMach*

Intention	: This predicate is true if and only if the machine assumes that some outdoor light sensors have a malfunction.
Scope	: mh
Prereferences	: pdref NF1 pdref NF2 pdref NF3
Postreferences	: msref Floor : M_F1 rsref Floor : R_F1

Domain Knowledge

Property *D_F1*

Formal	: $\Box(o435.neighborDoor[1] = o433.neighborDoor[1])$
NL	: The two rooms with room numbers 435 and 433 are connected by a door.
Prereferences	: akref Figure1

Property *D_F2*

Formal	: $\Box(o433.neighborDoor[2] = o431.neighborDoor[1])$
NL	: The two rooms with room numbers 433 and 431 are connected by a door.
Prereferences	: akref Figure1

Property *D_F3*

Formal	: $\Box(o425.neighborDoor[1] = o423.neighborDoor[1])$
NL	: The two rooms with room numbers 425 and 423 are connected by a door.
Prereferences	: akref Figure1

Property *D_F4*

Formal	: $\Box(o423.neighborDoor[2] = o421.neighborDoor[1])$
NL	: The two rooms with room numbers 423 and 421 are connected by a door.
Prereferences	: akref Figure1

Property *D_F5*

Formal	: $\Box(cl418.neighborDoor[2] = o416.neighborDoor[1])$
NL	: The two rooms with room numbers 418 and 416 are connected by a door.
Prereferences	: akref Figure1

Property *D_F6*

Formal	: $\Box(o416.neighborDoor[1] = o414.neighborDoor[1])$
NL	: The two rooms with room numbers 416 and 414 are connected by a door.
Prereferences	: akref Figure1

Property *D_F7*

Formal : $\Box(h1.westDoor = h2.eastDoor)$
NL : The eastern and middle hallway section are connected by a door.
Prereferences : akref Figure1

Property *D_F8*

Formal : $\Box(h2.westDoor = h3.eastDoor)$
NL : The middle and western hallway section are connected by a door.
Prereferences : akref Figure1

Requirement Specifications

Property *R_F1*

Formal : $\Box(\forall h \in HALLWAYS : (outLightSensorFaultMach \wedge h.occFM.occUsed \Rightarrow_{\leq TmalReact} h.light[1].sl.noMalSens.envEntity = 1))$
NL : Whenever some outdoor light sensors are not working correctly and the hallway is occupied for at least *TmalReact* 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.
Prereferences : pdref FM8

Property *R_F3*

NonFormal : The system provides reports on current and past energy consumption.
Prereferences : pdref FM9

Property *R_F4*

NonFormal : All malfunctions and unusual conditions are stored and reported on request.
Prereferences : pdref FM10

Property *R_F5*

NonFormal : All hardware connections have to be made according to DIN standards.
Prereferences : pdref NF6

Property *R_F6*

NonFormal : No hazardous condition for persons, inventory or building are allowed.
Prereferences : pdref NF7

Machine Specifications

Property M_F1

Formal : $\square(outLightSensorFaultMach \leftrightarrow ols1.malSens.malDetMach \lor ols2.malSens.malDetMach \lor ols3.malSens.malDetMach \lor ols4.malSens.malDetMach \lor ols5.malSens.malDetMach \lor ols6.malSens.malDetMach)$

NL : The machine assumes that some outdoor light sensor has a malfunction if and only if one of the six outdoor light sensors has a malfunction.

Property M_F2

Formal : $\square(\forall r \in ROOMS : (r.OLSMalFct \rightarrow r.curDefaultLightScene = r.malDefaultLightScene))$

NL : Whenever the machine assumes that some outdoor light sensor used by a room is not working correctly, then the default light scene is the default fault light scene, i.e., all ceiling lights are on.

Prereferences : pdref NF2

Property M_F3

Formal : $\square(indicatorMalFctEastStaircase.informCondMach \leftrightarrow sce.md.malSens.malDetMach)$

NL : The situation the facility manager should be informed about concerning the motion detector in the eastern staircase is given if and only if this motion detector has a malfunction.

Property M_F4

Formal : $\square(indicatorMalFctWestStaircase.informCondMach \leftrightarrow scw.md.malSens.malDetMach)$

NL : The situation the facility manager should be informed about concerning the motion detector in the western staircase is given if and only if this motion detector has a malfunction.

Property M_F5

Formal : $\Box(\forall r1 \in ROOMS : \forall r2 \in ROOMS : \forall h1 \in HALLWAYS : \forall h2 \in HALLWAYS :$
$(r1.CPFM.indicatorUnreasonInp.indicator = r2.CPFM.indicatorUnreasonInp.indicator \wedge$
$r1.CPFM.indicatorUnreasonInp.indicator = h1.CPFM.indicatorUnreasonInp.indicator \wedge$
$h1.CPFM.indicatorUnreasonInp.indicator = h2.CPFM.indicatorUnreasonInp.indicator))$

NL : The indicator light of the facility manager control panel concerning an unreasonable input is the same for all rooms and all hallways. Note that nevertheless the conditions for this indicator light differ from area to area.

Property M_F6

Formal : $\Box(o435.OLSMalFct \leftrightarrow ols1.malSens.malDetMach)$

NL : Room *o435* uses the following outdoor light sensors:

- outdoor light sensor *ols1*

Property M_F7

Formal : $\Box(o433.OLSMalFct \leftrightarrow ols1.malSens.malDetMach)$

NL : Room *o433* uses the following outdoor light sensors:

- outdoor light sensor *ols1*

Property M_F8

Formal : $\Box(o431.OLSMalFct \leftrightarrow ols1.malSens.malDetMach)$

NL : Room *o431* uses the following outdoor light sensors:

- outdoor light sensor *ols1*

Property M_F9

Formal : $\Box(o429.OLSMalFct \leftrightarrow ols1.malSens.malDetMach)$

NL : Room *o429* uses the following outdoor light sensors:

- outdoor light sensor *ols1*

Property *M_F10*

Formal : $\Box(m427.OLSMalFct \leftrightarrow ols1.malSens.malDetMach \vee ols5.malSens.malDetMach)$

NL : Room *m427* uses the following outdoor light sensors:

- outdoor light sensor *ols1*
- outdoor light sensor *ols5*

Property *M_F11*

Formal : $\Box(o425.OLSMalFct \leftrightarrow ols5.malSens.malDetMach)$

NL : Room *o427* uses the following outdoor light sensors:

- outdoor light sensor *ols5*

Property *M_F12*

Formal : $\Box(o423.OLSMalFct \leftrightarrow ols5.malSens.malDetMach)$

NL : Room *o423* uses the following outdoor light sensors:

- outdoor light sensor *ols5*

Property *M_F13*

Formal : $\Box(o421.OLSMalFct \leftrightarrow ols3.malSens.malDetMach \vee ols5.malSens.malDetMach)$

NL : Room *o421* uses the following outdoor light sensors:

- outdoor light sensor *ols3*
- outdoor light sensor *ols5*

Property *M_F14*

Formal : $\Box(o419.OLSMalFct \leftrightarrow ols3.malSens.malDetMach)$

NL : Room *o419* uses the following outdoor light sensors:

- outdoor light sensor *ols3*

Property *M_F15*

Formal : $\Box(o417.OLSMalFct \leftrightarrow ols3.malSens.malDetMach)$

NL : Room *o417* uses the following outdoor light sensors:

- outdoor light sensor *ols3*

Property *M_F16*

Formal : $\Box(o415.OLSMalFct \leftrightarrow ols3.malSens.malDetMach)$
NL : Room *o415* uses the following outdoor light sensors:

- outdoor light sensor *ols3*

Property *M_F17*

Formal : $\Box(hl411.OLSMalFct \leftrightarrow ols5.malSens.malDetMach)$
NL : Room *cl411* uses the following outdoor light sensors:

- outdoor light sensor *ols5*

Property *M_F18*

Formal : $\Box(cl426.OLSMalFct \leftrightarrow ols2.malSens.malDetMach)$
NL : Room *cl426* uses the following outdoor light sensors:

- outdoor light sensor *ols2*

Property *M_F19*

Formal : $\Box(o424.OLSMalFct \leftrightarrow ols2.malSens.malDetMach)$
NL : Room *o424* uses the following outdoor light sensors:

- outdoor light sensor *ols2*

Property *M_F20*

Formal : $\Box(cl422.OLSMalFct \leftrightarrow ols2.malSens.malDetMach \vee ols3.malSens.malDetMach)$
NL : Room *cl422* uses the following outdoor light sensors:

- outdoor light sensor *ols2*
- outdoor light sensor *ols3*

Property *M_F21*

Formal : $\Box(cl418.OLSMalFct \leftrightarrow ols4.malSens.malDetMach)$
NL : Room *cl418* uses the following outdoor light sensors:

- outdoor light sensor *ols4*

Property *M_F22*

Formal : $\Box(o416.OLSMalFct \leftrightarrow ols4.malSens.malDetMach)$
NL : Room *o416* uses the following outdoor light sensors:

- outdoor light sensor *ols4*

Property *M_F23*

Formal : $\Box(o414.OLSMalFct \leftrightarrow ols4.malSens.malDetMach)$
NL : Room *o414* uses the following outdoor light sensors:

- outdoor light sensor *ols4*

Property *M_F24*

Formal : $\Box(o412.OLSMalFct \leftrightarrow ols4.malSens.malDetMach \vee ols6.malSens.malDetMach)$
NL : Room *o412* uses the following outdoor light sensors:

- outdoor light sensor *ols4*
- outdoor light sensor *ols6*

Property *M_F25*

Formal : $\Box(hl410.OLSMalFct \leftrightarrow ols6.malSens.malDetMach)$
NL : Room *cl410* uses the following outdoor light sensors:

- outdoor light sensor *ols6*

Index