Problem Specification 3.0

Project: Light_Heating_32_4

Authors: Kronenburg, Peper

Date: December 15, 1998

Referenced Document: BD : team3.2-BuildingDescription

Referenced Document: PD : team3.2-ProblemDescription

Referenced Document: DIC : team3.2-Dictionary

Contents

Introduction

Description Classes

Appendix

1 Content

This *problem specification* contains a specification for a new light and heating control system for the 4^{th} floor of building 32 of the University of Kaiserslautern. The focus is on a specification of the behaviour of the control system. The specification consists of a *domain knowledge* part and a *requirements and specification* part.

The domain knowledge part describes the behaviour of the *environment*, i.e. the behaviour of the system as it is *without* any control by a *machine*. The requirements and specification part describes the behaviour of the system as it should be *when* controlled by a machine. Thus, a *system* is divided into two parts: an *environment* and a *machine*. This classification is based on [ZJ97].

The input documents for this problem specification have been:

team3.2-ProblemDescription This document contains an informal description of *needs*. These needs describe in natural language the *requirements* concerning the system.

team3.2-BuildingDescription This document contains an informal description of the installation of the 4^{th} floor of building 32 of the University of Kaiserslautern. Hence, this document concentrates on the *domain knowledge* concerning the system.

team3.2-Dictionary This dictionary contains explanations of several terms used in the two documents team3.2-ProblemDescription and team3.2-BuildingDescription.

Customer Review Results These documents contain the results of several discussions with the customer in order to clarify misunderstandings, inconsistencies, and omissions during the production of this problem specification.

2 Structure

To present the problem specification in a concise and intelligible way, it is structured by several concepts. The problem specification consists of several *description classes*. Each description class deals with a specific *aspect* concerning the system. Such an aspect can be for example the temperature control in a room or the behaviour of a sensor in the case that it has no malfunction. The possible parts of a description class are introduced in the following sections:

2.1 Formal Parameters

Parameterization of description classes is supported for two kinds of parameters: *Domain parameters* and *number parameters* (if they are used at all in a description class) are introduced at the beginning after the keyword FORMAL PARAMETERS.

A *domain parameter* is indicated by the keyword ***Domain***. It represents a set of values that has to be fixed when objects of this class are instantiated (for example the set of values that can be delivered by a sensor). A *number parameter* is indicated by the keyword ***Timed Function***. It represents a natural number that has to be fixed at the instantiation this class (for example the number of window sections in a room).

2.2 Base Classes

A description class may be refined into a more specific class (*inheritance*). If a description class R is a refinement of a base class B this is expressed at the beginning of class R after the keyword BASE CLASSES. The meaning of such a refinement is that all *entities* and *properties* introduced in class B are also available in the refined class R. (Entities and properties are explained below.)

2.3 Signature

The SIGNATURE part contains an (*aggregation*) of class instantiations and other entities. All entities are only visible in this class or in derived classes.

2.3.1 Entities

Entities are used to describe *properties* in the DOMAIN KNOWLEDGE and REQUIREMENTS/SPECIFICATIONS part. The following entity types may be used:

Object An object is the instantiation of a description class.

Domain A domain represents a set of values.

Timed Function A timed function is a function where the produced value depends on the time. For example, a timed function is used to model the current temperature in a room.

Timed Predicate A timed predicate is a predicate where the validity depends on the time. For example, a timed predicate is used to model whether a room is occupied or not.

Function The value produced by an (untimed) function does not depend on the time. The usual addition of two numbers is an example for such a time independent function.

Predicate The validity of an (untimed) predicate does not depend on the time. Time independent predicates are used for example to model default values of timed predicates.

Time Constant A time constant is a special (untimed) function representing a constant of the domain TIME. Since time is very important for the specification of the behaviour of a system these constants have a specific keyword.

2.3.2 Entity attributes

For each entity a formal definition is given after the corresponding keyword. Additionally, an explanation in natural language is given after the keyword **Intention**. Furthermore, the **Scope** of an entity is specified. As scope of an entity one of the following four terms can be used.

environment controlled, not visible for the machine This means that the entity is controlled by the environment and is not visible for the machine. An example is the temperature of a room in the real world.

environment controlled, visible for the machine This means that the entity is controlled by the environment and is visible for the machine. An example is the value delivered by a temperature sensor.

machine controlled, visible for the environment This means that the entity is controlled by the machine and is visible for the environment. An example is the value determined for the setting of an actuator.

machine controlled, not visible for the environment This means that the entity is controlled by the machine and is not visible for the environment. An example is the value computed by the machine concerning the occupancy of a room.

Since entities of this scope should normally not be part of a requirement or a specification, in most cases an additional comment *This entity is explicitly introduced by the customer.* is given. If such a comment is missed an entity of this scope has only been introduced as an abbreviation of a more complex expression.

For all entities an additional **Comment** can be made.

If an entity is not introduced as an abbreviation of a more complex expression references are mentioned . These **References** point to the parts of the input documents where this entity has been introduced.

2.4 Domain Knowledge, Requirements, Specifications

As far as the behaviour of the system is considered the domain knowledge as well as the requirements and specifications part of a description class contain a set of behaviour properties.

Each **Property** is either given as **Formal** using a temporal logic [KGP96] or as **Non-Formal**, if it can not be formalized appropriately. For each formal expression a description in natural language (**NL**) is given. As for entities, references and comments are further possible attributes.

2.5 Global Signature

Before any description class is specified a *Global Signature* is introduced. It contains entities (especially domains) that can be used by all following description classes.

Besides these globally introduced entities, the following (built-in) domains can be used with the obvious meaning: NAT, INTEGER, REAL. Furthermore, the domain TIME contains all possible time values. Milliseconds are used as time unit.

2.6 Main Description Class

The final problem specification is contained in the description class termed *Main Description Class*, usually specified at the end of the document.

2.7 Further Remarks

To support the navigation and understanding of this problem specification, it contains additionally after this introduction

- a table of contents
- several figures representing the inheritance hierarchy
- several figures representing the aggregation hierarchy

and at the end of this document an index.

References

[KGP96] Martin Kronenburg, Reinhard Gotzhein, and Christian Peper. A tailored real time temporal logic for specifying requirements of building automation systems. SFB 501 Bericht 16/96, Sonderforschungsbereich 501, Fachbereich Informatik, Universität Kaiserslautern, 1996.

[ZJ97] Pamela Zave and Michael Jackson. Four dark corners of requirements engineering. *IEEE Transactions on Software Engineering and Methodology*, 6(1):1–30, January 1997.

Global Signature

Domain $TEMP = \{-40 \ldots 70\} < INTEGER$

Intention	: This domain contains the possible temperature values in degrees Celsius.
References	: BD/Sensors: room temperature sensor (rts), CustomerReviewResults

Domain $LIGHT\_VALUES = \{1 \ldots 10000\} < INTEGER$

Intention	: This domain contains the possible light intensity values in lux.
References	: BD/Sensors: outdoor light sensor (ols), CustomerReviewResults

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

Intention	: This domain contains the values 0 and 1.
References	: BD/Sensors: pushbutton (pb), BD/Sensors: door closed contact (dcc), BD/Sensors: door open contact (doc), BD/Sensors: door lock contact (dlc), BD/Sensors: motion detector (imd), BD/Actuators: dimmable light (sll), BD/Actuators: dimmable light (cia)

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

Intention	: This domain contains the values 0 to 100.
References	: BD/Actuators: blind lifter/rotator, BD/Actuators: dimmable light (dll), BD/Actuators: radiator valve

Domain $DT1\_RANGE < TIME$

Intention	: This domain contains the values that can be used for a time delay dt_1.
References	: PD/Paragraph 7

Domain $DT2\_RANGE < TIME$

Intention	: This domain contains the values that can be used for a time delay dt_2.
References	: PD/Paragraph 7

Domain $DT3\_RANGE < TIME$

Intention	: This domain contains the values that can be used for a time delay dt_3.
References	: PD/Paragraph 7

Domain $COMFORT\_TEMP < TEMP$

Intention	: This domain contains the values that can be used for a comfort temperature.
References	: PD/UH1
Comment	: Note that this domain is not specified by the facility manager but is predefined, containing all possible comfort temperatures, not only those allowed by the facility manager.

Domain *WORKING_TIME_RANGE*

Intention	:	This domain contains the values that can be used for a working time.
References	:	PD/UH4
Comment	:	Note that this domain is not specified by the facility manager but is predefined, containing all possible working times, not only those allowed by the facility manager.

Domain *HOLIDAY_TIME_RANGE*

Intention	:	This domain contains the values that can be used for a holiday time.
References	:	PD/UH5
Comment	:	Note that this domain is not specified by the facility manager but is predefined, containing all possible holiday times, not only those allowed by the facility manager.

Domain *CALENDAR_TIME*

Intention	:	This domain contains the values that can be used for a calendar time, i.e. a date.
References	:	PD/FMH4, PD/FMH5
Comment	:	Note that this domain changes from day to day, since it consists always of the days of the next year

Domain $THREE\_PT\_ACT\_RANGE = \{FORWARD, BACKWARD, STOP\}$

Intention	:	This domain contains the possible control states of a three point actuator.
References	:	BD/Actuators: introduction

Domain *STRING*

Intention	:	This domain contains the values that can be entered as a string.
References	:	CustomerReviewResults

Timed Function $now() \rightarrow TIME$

Intention	:	This function represents the current time, i.e. the current day, month, year, hour, minute, second, millisecond, ...
Scope	:	environment controlled, not visible for the machine
References	:	PD/UH4, PD/UH5
Comment	:	Appropriate functions providing for example only the date or the current daytime should be provided. But this is a task of the implementation.

Timed Predicate *daytime*

Intention	:	This predicate is true during daytime, otherwise it is false.
Scope	:	environment controlled, not visible for the machine
References	:	PD/Paragraph 4, PD/NFH4, CustomerReviewResults
Comment	:	*Daytime* is the time between 6 o'clock in the morning and 8 o'clock in the evening.

Description Class EnteredValue

Intention: This class provides all definitions to describe the entering of a value using an appropriate device (in our case this device is the 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*. c is the value which is *nearest* to e. c is the value used in all properties and not the originally entered value e.

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

Formal Parameters

Domain *CHECKED_DOM*

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

Signature

Timed Function $entered() \rightarrow STRING$

Intention	: This function represents the value entered by a person.
Scope	: environment controlled, not visible for the machine
References	: PD/UH1, PD/UH3, PD/UH4, PD/UH5, PD/UH6, PD/FMH3, PD/FMH4, PD/FMH5, PD/FMH6, PD/NFH4, CustomerReviewResults

Timed Function $checked() \rightarrow CHECKED\_DOM$

Intention	: This function represents the value decided by the machine to use.
Scope	: machine controlled, not visible for the environment
References	: PD/UH1, PD/UH4, PD/FMH3
Comment	: *checked* is determined using the function *nearestValue*. This entity is explicitly introduced by the customer.

Timed Function $nearestValue : (STRING) \rightarrow CHECKED\_DOM$

Intention	: This function returns the value which is the *nearest* one in the domain CHECKED_DOM w.r.t. the first argument.
Scope	: machine controlled, not visible for the environment
References	: PD/UH1, PD/UH4, PD/FMH3
Comment	: This entity is explicitly introduced by the customer. The specification of this function for an object of this class has to be provided by the domain expert.

Timed Function $checkTime : () \rightarrow Time$

Intention	: This function represents the time needed to check a value entered by a person using the function *nearestValue*.
Scope	: environment controlled, not visible for the machine
Comment	: The specification of this function for an object of this class has to be provided by the domain expert.

Requirements/Specifications

Property P_EV_1

Formal	: $\Box(entered \rhd^{nearestValue}_{\leq checkTime} checked)$
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*.
References	: PD/UH1, PD/UH4, PD/FMH3

Description Class DynamicInterval

Intention: This class provides all definitions to describe a *dynamic interval. Dynamic* means that the values of the start and end point of the interval can change over time. These values can be entered by a user. Therefore the start and end point are objects of the class EnteredValue.

Formal Parameters

Domain INT_DOM

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

Signature

Object $start : EnteredValue(CHECKED\_DOM = INT\_DOM)$

Intention	: This object represents the starting point of the interval.
Scope	: environment controlled, not visible for the machine
References	: PD/Paragraph 3, PD/UH4, PD/UH5, PD/FMH3, PD/FMH4

Object $end : EnteredValue(CHECKED\_DOM = INT\_DOM)$

Intention	: This object represents the end point of the interval.
Scope	: environment controlled, not visible for the machine
References	: PD/Paragraph 3, PD/UH4, PD/UH5, PD/FMH3, PD/FMH4

Predicate $leq : (INT\_DOM, INT\_DOM)$

Intention	: This predicate is true iff the first argument is *less* than or equal to the second argument.
Scope	: machine controlled, not visible for the environment
References	: CustomerReviewResults

Requirements/Specifications

Property P_DI_1

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

Description Class CalendarInterval

Intention: This class provides all definitions to describe an interval of calendar times. The start and end point of the interval can change over time. Additionally a timed predicate indicates whether the current date is in this interval or not.

Comment: Since it is a general class, no references are given here.

Base Classes

Class $DynamicInterval(INT\_DOM = CALENDAR\_TIME)$

Intention	:	All definitions of a dynamic interval are also provided here, using the domain containing the calendar times as interval domain.
References	:	PD/Paragraph 3, PD/UH5, PD/FMH4

Signature

Timed Predicate *actual*

Intention	:	This predicate is true iff the current date is in the calendar interval.
Scope	:	machine controlled, not visible for the environment
References	:	PD/Paragraph 3, PD/UH5, PD/FMH4
Comment	:	This entity is explicitly introduced by the customer.

Requirements/Specifications

Property P_CI_1

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

Description Class ControlledInput

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

Formal Parameters

Domain *INPUT_DOM*

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

Signature

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

Intention	: This object represents the range of possible user entries.
Scope	: environment controlled, not visible for the machine
References	: PD/NF10, PD/UH1, PD/UH4, PD/UH5, PD/FMH3, PD/FMH10
Comment	: This range is specified by the facility manager.

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

Intention	: This object represents the default value given by the facility manager.
Scope	: environment controlled, not visible for the machine
References	: PD/UH1, PD/UH4, PD/UH5, PD/FMH3
Comment	: Note that by this declaration it is only guaranteed that the default value is an element of the domain *INPUT_DOM*. It is not guaranteed that it is a value of the interval specified by the facility manager. This is expressed by the property P_CIN_3 of this class. For each object of the class *ControlledInput* an initial value for *default.checked* has to be specified.

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

Intention	: This object represents the value entered by a user.
Scope	: environment controlled, not visible for the machine
References	: PD/UH1, PD/UH4, PD/UH5, PD/UH6
Comment	: Note that by this declaration it is only guaranteed that the entered value is an element of the domain *INPUT_DOM*. It is not guaranteed that it is a value of the interval specified by the facility manager. This is expressed by the property P_CIN_2 of this class. For each object of the class *ControlledInput* an initial value for *default.checked* has to be specified.

Timed Function $value() \rightarrow INPUT\_DOM$

Intention	: This function represents the currently valid value. It depends on the inputs of the facility manager and the input of a user.
Scope	: machine controlled, not visible for the environment
References	: PD/UH1, PD/UH4, PD/UH5, PD/FMH3
Comment	: This entity is explicitly introduced by the customer.

Timed Predicate *userValueEntered*

Intention	: This predicate is true iff the user has entered a value.
Scope	: machine controlled, not visible for the environment
References	: PD/UH1, PD/UH4, PD/UH5, PD/UH6, PD/FMH3
Comment	: This entity is explicitly introduced by the customer.

Requirements/Specifications

Initialization

Property I_CIN_1

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

Property I_CIN_2

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

End of Initialization

Property P_CIN_1

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

Property P_CIN_2

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

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

References : PD/UH1, PD/UH4, PD/UH5, PD/FMH3

Property P_CIN_3

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

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

References : PD/FMH10, CustomerReviewResults

Description Class NoMalfunctionSensor

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

It is distinguished between three values. The value as it exists in the real world (which can never be visible for the machine), the analog measured value of this real world value, and the digital measured value which is a conversion of the analog measured value.

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

Formal Parameters

Domain *ENV_DOMAIN*

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

Domain *MEASURED_DOMAIN*

Intention	: This domain contains the possible analogously measured values.

Domain *CONVERTED_DOMAIN*

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

Signature

Timed Function $envEntity() \rightarrow ENV\_DOMAIN$

Intention	: This function represents the value of the measured entity in the real world.
Scope	: environment controlled, not visible for the machine
References	: BD/Sensors: introduction
Comment	: Note that this function represents the situation as it is in the real world and not how it is seen by the machine.

Timed Function $measuredEntity() \rightarrow MEASURED\_DOMAIN$

Intention	: This function represents the analogously measured value of the entity of the environment.
Scope	: environment controlled, not visible for the machine
References	: BD/Sensors: introduction

Timed Function $convertedEntity() \rightarrow CONVERTED\_DOMAIN$

Intention	: This function represents the digitally converted value of the measured value.
Scope	: environment controlled, visible for the machine
References	: BD/Sensors:introduction

Timed Function $modifyReaction(ENV\_DOMAIN) \rightarrow MEASURED\_DOMAIN$

Intention	: This function represents the way in which a value in the real world (*envEntity*) is modified during the reaction of the sensor resulting in an analogously measured value *measuredEntity*.
Scope	: environment controlled, not visible for the machine
References	: BD/Sensors:introduction

Timed Function $modifyConversion(MEASURED\_DOMAIN) \rightarrow CONVERTED\_DOMAIN$

Intention	: This function represents the way in which an analogously measured value *measuredEntity* is modified during the conversion into a digital value (*convertedEntity*).
References	: BD/Sensors:introduction

Time Constant *reactionTime*

Intention	: This time constant represents the *reaction time* of the sensor.
Scope	: environment controlled, not visible for the machine
References	: BD/Sensors: introduction
Comment	: 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.

Time Constant *conversionTime*

Intention	: This time constant represents the *conversion time* of the sensor.
References	: BD/Sensors: introduction
Comment	: The *conversion time* is the time needed to convert an analogously measured value (*measuredEntity*) into the corresponding digital value (*convertedEntity*).

Domain Knowledge

Property D_NMS_1

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

Property D_NMS_2

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

Description Class MalfunctionSensor

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

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

Comment: Since it is a general class, no explicit references are given here.

Formal Parameters

Domain *MAL_CONVERTED_DOMAIN*

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

Signature

Timed Predicate *malDetEnv*

Intention	: This Predicate is true iff the sensor existing in the real world has a malfunction.
Scope	: environment controlled, not visible for the machine
References	: PD/U11, PD/FM7, PD/FM8, PD/NF2, PD/NF3, PD/NF4, PD/NF5, PD/FMH12, PD/NFH1, PD/NFH3, PD/NFH4
Comment	: Note that this predicate represents the value as it exists in the real world. This value is not visible for the machine. It can be used to express requirements.

Timed Predicate *malDetMach*

Intention	: This Predicate is true iff the machine assumes that the sensor existing in the real world has a malfunction.
Scope	: machine controlled, not visible for the environment
References	: PD/U11, PD/FM7, PD/FM8, PD/NF2, PD/NF3, PD/NF4, PD/NF5, PD/FMH12, PD/NFH1, PD/NFH3, PD/NFH4
Comment	: Note that this predicate represents the value as it seen by the machine. This entity is explicitly introduced by the customer.

Object *manualEnteredVal* : *EnteredValue*
($CHECKED\_DOM = MAL\_CONVERTED\_DOMAIN$)

Intention	: This object represents the value entered manually by a user in the case of a malfunction of this sensor.
Scope	: environment controlled, not visible for the machine
References	: PD/NFH1, PD/NFH9

Timed Predicate *stateManualEnteredValue*

Intention	: This predicate is true iff a value, that should be used in the case of a malfunction of the sensor, has been entered manually by a user.
Scope	: machine controlled, not visible for the environment
References	: PD/NFH1, PD/NFH9
Comment	: This entity is explicitly introduced by the customer.

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

Intention	: This function represents the value of the sensor determined automatically by the machine in the case of a malfunction of the sensor.
Scope	: machine controlled, not visible for the environment
References	: PD/NF2, PD/NF3, PD/NF4, PD/NF5, PD/NFH1, PD/NFH3, PD/NFH9
Comment	: The specification of this function for an object of this class has to be provided by the domain expert.

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

Intention	: This function represents the value of the sensor used by the machine in the case of a malfunction of the sensor.
Scope	: machine controlled, not visible for the environment
References	: PD/NF2, PD/NF3, PD/NF4, PD/NF5, PD/NF1, PD/NFH1, PD/NFH3, PD/NFH9

Timed Predicate *FMInformed*

Intention	: This predicate is true iff the facility manager is informed about an *important event* concerning this sensor. An *important event* is either the detection of a malfunction of the sensor or that a malfunction of the sensor has been repaired.
Scope	: environment controlled, not visible for the machine
References	: CustomerReviewResults, PD/FM7, PD/FMH7
Comment	: This has to be established by sending an email.

Timed Predicate *userInformed*

Intention	: This predicate is true iff the user is informed about an *important event* concerning this sensor. An *important event* is either the detection of a malfunction or that a malfunction has been repaired.
Scope	: environment controlled, not visible for the machine
References	: CustomerReviewResults, PD/U11
Comment	: This has to be established by sending an email.

Time Constant $T_{infDelay}$

Intention : This time constant represents the maximal delay between an event concerning this sensor a user or the facility manager has to be informed about (e.g. a malfunction or the repair of a malfunction) and the time point the user or the facility manager is informed.

References : CustomerReviewResults, PD/FM7, PD/FMH7, PD/U11

Time Constant T_{info}

Intention : This time constant represents the maximal time span a user or the facility manager is informed.

References : CustomerReviewResults, PD/FM7, PD/FMH7, PD/U11

Requirements/Specifications

Initialization

Property I_MS_1

Formal : $\neg stateManualEnteredValue$

NL : Initially no value for the case of a malfunction is entered manually by a user.

References : CustomerReviewResults

Property I_MS_2

Formal : $\neg userInformed$

NL : Initially the user is not informed about a malfunction of this sensor.

References : CustomerReviewResults

Property I_MS_3

Formal : $\neg FMInformed$

NL : Initially the facility manager is not informed about a malfunction of this sensor.

References : CustomerReviewResults

End of Initialization

Property P_MS_1

Formal : $\Box(stateManualEnteredValue \rightarrow (valueMal = manualEnteredVal.checked) \wedge (\neg stateManualEnteredValue \rightarrow (valueMal = automaticDetermValue)))$

NL : If a value for the status of the sensor in the case of a malfunction has been entered manually this is the value used by the machine in the case of a malfunction of the sensor.
Otherwise, the value used by the machine in the case of a malfunction of the sensor is the value determined automatically in the case of a malfunction of the sensor.

References : CustomerReviewResults, PD/NFH1, PD/NFH9

Property P_MS_2

Formal : $\Box([userInformed] \rightarrow \Diamond_{\leq T_{info}} \neg userInformed)$

NL : Whenever a user is informed the predicate userInformed becomes false again within T_{info} time units.

References : CustomerReviewResults

Comment : By this requirement it is expressed, that the user is not permanently informed, but that it is a kind of signal, e.g. an email.

Property P_MS_3

Formal : $\Box([FMInformed] \rightarrow \Diamond_{\leq T_{info}} \neg FMInformed)$

NL : Whenever the facility manager is informed the predicate FMInformed becomes false again within T_{info} time units.

References : CustomerReviewResults

Comment : By this requirement it is expressed, that the facility manager is not permanently informed, but that it is a kind of signal, e.g. an email.

Property P_MS_4

Formal : $\Box([malDetMach] \rightarrow \Diamond_{\leq T_{infDelay}} userInformed)$

NL : Whenever the machine assumes that a malfunction of the sensor has occurred, the user is informed about this within $T_{infDelay}$ milliseconds.

References : CustomerReviewResults, PD/U11

Property P_MS_5

Formal : $\Box([malDetMach] \rightarrow \Diamond_{\leq T_{infDelay}} FMInformed)$

NL : Whenever the machine assumes that a malfunction of the sensor has occurred, the facility manager is informed about this within $T_{infDelay}$ milliseconds.

References : CustomerReviewResults, PD/FM7, PD/FMH7

Property P_MS_6

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

Property P_MS_7

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

Description Class Sensor

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

Signature

Object *noMalSens* : *NoMalfunctionSensor*

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

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

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

Timed Predicate *stateManualEnteredMal*

Intention	: This predicate is true iff it has been manually entered whether there is a malfunction of the sensor or whether there is no malfunction of it.
Scope	: machine controlled, not visible for the environment
References	: CustomerReviewResults, PD/FM11, PD/FMH13
Comment	: This entity is explicitly introduced by the customer.

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

Intention	: This object represents a value entered manually by a user concerning a malfunction of the sensor. 1 stands for a malfunction, 0 stands for no malfunction.
Scope	: environment controlled, not visible for the machine
References	: PD/FM11, PD/FMH13

Timed Predicate *automaticDetMal*

Intention	: This predicate is true iff a malfunction of the sensor has been detected automatically, i.e. not by entering it manually.
Scope	: machine controlled, not visible for the environment
References	: CustomerReviewResults
Comment	: The specification of this function for an object of this class has to be provided by the domain expert.

Timed Function $usedValue() \to CONVERTED\_DOMAIN$

Intention	: This function models the current value used by the machine as the value of the sensor. This depends on whether there is a malfunction of the sensor or not.
Scope	: machine controlled, not visible for the environment
References	: PD/NF2, PD/NF3, PD/NF4, PD/NF5, PD/NF1, PD/NFH1, PD/NFH3, PD/NFH9
Comment	: This entity is explicitly introduced by the customer.

Requirements/Specifications

Initialization

Property I_S_1

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

End of Initialization

Property P_S_1

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

NL : The machine assumes a malfunction of the sensor iff one of the following conditions is fulfilled:

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

References : CustomerReviewResults

Property P_S_2

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

Property P_S_3

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

Description Class RoomTempSensor

Intention: This class provides all definitions to describe the behavior of a *room temperature sensor.*

Base Classes

Class $Sensor(ENV\_DOMAIN = REAL, MEASURED\_DOMAIN = REAL, CONVERTED\_DOMAIN = REAL)$

Intention	:	All definitions of Sensor are also provided here using the real numbers as the possible values of the sensor existing in the real world and using the real numbers as the possible analogously measured values and using the real numbers as the possible digitally converted measured values.
References	:	CustomerReviewResults

Signature

Domain $ROOM\_TEMP = \{-40 \ldots 70\} < INTEGER$

Intention	:	This domain contains the possible values measured by a room temperature sensor.
References	:	BD/Sensors: room temperature sensor (rts)

Domain Knowledge

Property D_RTS_1

Formal	:	$\square(noMalSens.conversionTime = 160000)$
NL	:	The conversion time of a room temperature sensor is 160 seconds.
References	:	BD/Sensors: room temperature sensor (rts)
Comment	:	Note that the time unit is milliseconds.

Property D_RTS_2

Formal	:	$\square(noMalSens.reactionTime = 10000)$
NL	:	The reaction time of a room temperature sensor is 10 seconds.
References	:	BD/Sensors: room temperature sensor (rts)
Comment	:	Note that the time unit is milliseconds.

Description Class OutdoorTempSensor

Intention: This class provides all definitions to describe the behavior of a *outdoor temperature sensor.*

Base Classes

Class $Sensor(ENV\_DOMAIN = REAL, MEASURED\_DOMAIN = REAL,$
$CONVERTED\_DOMAIN = OUT\_TEMP)$

Intention	: All definitions of Sensor are also provided here using the real numbers as the possible values of the sensor existing in the real world and using the real numbers as the possible analogously measured values and using the real numbers as the possible digitally converted measured values.
References	: CustomerReviewResults

Signature

Domain $OUT\_TEMP = \{-40 \dots 60\} < NAT$

Intention	: This domain contains the possible values measured by an outdoor temperature sensor.
References	: BD/Sensors: outdoor temperature sensor (ots)

Domain Knowledge

Property D_OTS_1

Formal	: $\Box(noMalSens.conversionTime = 160000)$
NL	: The conversion time of an outdoor temperature sensor is 160 seconds.
References	: BD/Sensors: outdoor temperature sensor (ots)
Comment	: Note that the time unit is milliseconds.

Property D_OTS_2

Formal	: $\Box(noMalSens.reactionTime = 120000)$
NL	: The reaction time of an outdoor temperature sensor is 120 seconds.
References	: BD/Sensors: outdoor temperature sensor (ots)
Comment	: Note that the time unit is milliseconds.

Description Class OutdoorLightSensor

Intention: This class provides all definitions to describe the behavior of a *outdoor light sensor*.

Base Classes

Class $Sensor(ENV\_DOMAIN = REAL, MEASURED\_DOMAIN = REAL, CONVERTED\_DOMAIN = OUT\_LIGHT\_VALUES)$

Intention	:	All definitions of Sensor are also provided here using the real numbers as the possible values of the sensor existing in the real world and using the real numbers as the possible analogously measured values and using the real numbers as the possible digitally converted measured values.
References	:	CustomerReviewResults

Signature

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

Intention	:	This domain contains the possible values measured by an outdoor light sensor.
References	:	BD/Sensors: outdoor light sensor (ols)

Timed Function $lastCorrectOutLight() \rightarrow OUT\_LIGHT\_VALUES$

Intention	:	This function represents the last correctly measured value of the outdoor light intensity.
Scope	:	environment controlled, not visible for the machine
References	:	PD/NF2
Comment	:	The specification of this function for an object of this class has to be provided by the domain expert.

Domain Knowledge

Property D_OLS_1

Formal	:	$\Box(noMalSens.conversionTime = 1000)$
NL	:	The conversion time of an outdoor light sensor is 1 second.
References	:	BD/Sensors: outdoor light sensor (ols)
Comment	:	Note that the time unit is milliseconds.

Property D_OLS_2

Formal	:	$\Box(noMalSens.reactionTime = 10)$
NL	:	The reaction time of a water temperature sensor is 10 milliseconds.
References	:	BD/Sensors: outdoor light sensor (ols)

Requirements/Specifications

Property P_OLS_1

Formal	: $\Box(malSens.automaticDetermValue = lastCorrectOutLight)$
NL	: The value of the sensor determined automatically in the case of a malfunction of the outdoor light sensor corresponds with the last correctly measured value.
References	: PD/NF2

Description Class WaterTempSensor

Intention: This class provides all definitions to describe the behavior of a *water temperature sensor*.

Base Classes

Class $Sensor(ENV\_DOMAIN = REAL, MEASURED\_DOMAIN = REAL, CONVERTED\_DOMAIN = WATER\_TEMP)$

Intention	: All definitions of Sensor are also provided here using the real numbers as the possible values of the sensor existing in the real world and using the real numbers as the possible analogously measured values and using the real numbers as the possible digitally converted measured values.
References	: CustomerReviewResults

Signature

Domain $WATER\_TEMP = \{0 \ldots 100\} < NAT$

Intention	: This domain contains the possible values measured by a water temperature sensor.
References	: BD/Sensors: water temperature sensor (wts)

Domain Knowledge

Property D_WTS_1

Formal	: $\Box(noMalSens.conversionTime = 160000)$
NL	: The conversion time of a water temperature sensor is 160 seconds.
References	: BD/Sensors: water temperature sensor (wts)
Comment	: Note that the time unit is milliseconds.

Property D_WTS_2

Formal	: $\Box(noMalSens.reactionTime = 100000)$
NL	: The reaction time of a water temperature sensor is 120 seconds.
References	: BD/Sensors: water temperature sensor (wts)
Comment	: Note that the time unit is milliseconds.

Description Class BinarySensor

Intention: This class provides all definitions to describe the behavior of a *binary sensor*. A binary sensor is a sensor that can return a 0 or a 1.

Comment: Since it is a general class, no references are given here.

Base Classes

Class $Sensor(ENV\_DOMAIN = BINARY, MEASURED\_DOMAIN = BINARY, CONVERTED\_DOMAIN = BINARY)$

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

Domain Knowledge

Property D_BS_1

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

Property D_BS_2

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

Property D_BS_3

Formal	: $\Box(\forall b \in BINARY : noMalSens.modifyConversion(b) = b)$
NL	: For a binary sensor the function representing the way in which an analogously measured value (*measuredEntity*) is modified during the conversion into a digital value (*convertedEntity*) is the identity.
References	: CustomerReviewResults
Comment	: Note that no conversion is needed.

Description Class Light

Intention: This class provides all definitions that are relevant for the description of a light.

Base Classes

Class *BinarySensor*

Intention	:	All definitions of a binary sensor are also provided here, i.e., the status line of the light is regarded as a *s*ensored value.
References	:	BD/Sensors: dimmable light (sll), BD/Actuators: figure, BD/Offices
Comment	:	The function *BinarySensor::convertedEntity* represents the status line of the light.

Domain Knowledge

Property D_L_1

Formal	:	$\Box(noMalSens.reactionTime = 10)$
NL	:	The reaction time of a status line is 10 ms.
References	:	BD/Actuators: dimmable light (sll)
Comment	:	Note that this sensor is describe in the actuator list in the document *Building Description.*

Description Class DoorContact

Intention: This class provides all definitions to describe the behavior of the different kinds of door contacts. These different kinds are *door closed contact*, *door open contact*, and *door lock contact*.

Base Classes

Class *BinarySensor*

Intention	: All definitions of BinarySensor are also provided here.

Domain Knowledge

Property D_DC_1

Formal	: $\Box(noMalSens.reactionTime = 10)$
NL	: The reaction time of a door contact is 10 ms.
References	: BD/Sensors: door closed contact (dcc), BD/Sensors: door open contact (doc), BD/Sensors: door lock contact (dlc)

Description Class MotDet

Intention: This class provides all definitions to describe the behavior of a motion detector.

Base Classes

Class *BinarySensor*

Intention	: All definitions of BinarySensor are also provided here.

Description Class NoMalfunctionActuator

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

It is distinguished between two values. The value as it exists in the real world (envEntity) (which can never be visible for the machine or controlled directly by the machine) and the value adjusted by the machine (adjustedEntity).

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

Formal Parameters

Domain *ENV_DOMAIN*

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

Domain *ADJUSTED_DOMAIN*

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

Signature

Timed Function $envEntity() \rightarrow ENV\_DOMAIN$

Intention	: This function represents the value of the actuator existing in the real world.
Scope	: environment controlled, not visible for the machine
References	: BD/Actuators: introduction
Comment	: 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.

Timed Function $adjustedEntity() \rightarrow ADJUSTED\_DOMAIN$

Intention	: This function represents the value that is adjusted by the machine.
Scope	: machine controlled, visible for the machine
References	: BD/Actuators: introduction

Time Constant *reactionTime*

Intention	: This time constant represents the *reaction time* of the actuator.
Scope	: environment controlled, not visible for the machine
References	: BD/Actuators: introduction
Comment	: The *reaction time* is the time between a change of the value adjusted by the machine (*adjustedEntity*) and the time point the value of the actuator in the real world (*envEntity*) changes its value *correspondingly.*

Timed Function $modifyReaction(ADJUSTED\_DOMAIN) \rightarrow ENV\_DOMAIN$

Intention	: This function represents the way in which a value adjusted by the machine (*adjustedEntity*) is modified during the reaction of the actuator resulting in the value of the entity in the real world (*envEntity*).
Scope	: environment controlled, not visible for the machine
References	: BD/Actuators: introduction
Comment	: The specification of this function for an object of this class has to be provided by the domain expert.

Description Class MalfunctionActuator

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

Signature

Timed Predicate *malDetEnv*

Intention	: This predicate is true iff the actuator existing in the real world has a malfunction.
Scope	: environment controlled, not visible for the machine
References	: CustomerReviewResults
Comment	: Note that this value is not visible for the machine. It can be used to express requirements.

Timed Predicate *malDetMach*

Intention	: This Predicate is true iff the machine assumes that the sensor existing in the real world has a malfunction.
Scope	: machine controlled, not visible for the environment
References	: CustomerReviewResults

Timed Predicate *FMInformed*

Intention	: This predicate is true iff the facility manager is informed about an *important event* concerning this actuator.
Scope	: environment controlled, not visible for the machine
References	: CustomerReviewResults, PD/U11
Comment	: This has to be established by sending an email. An *important event* is either the detection of a malfunction of the actuator or that a malfunction of the actuator has been repaired.

Timed Predicate *userInformed*

Intention	: This predicate is true iff the user is informed about an *important event* concerning this actuator.
Scope	: environment controlled, not visible for the machine
References	: CustomerReviewResults, PD/U11
Comment	: This has to be established by sending an email. An *important event* is either the detection of a malfunction of the actuator or that a malfunction of the actuator has been repaired.

Time Constant $T_{infDelay}$

Intention	: This time constant represents the maximal delay between an event concerning this actuator a user or the facility manager has to be informed about (e.g. a malfunction or the repair of a malfunction) and the time point the user or the facility manager is informed.

Time Constant T_{info}

Intention	: This time constant represents the maximal time span a user or the facility manager is informed.

Requirements/Specifications

Initialization

Property I_MA_1

Formal	: $\neg userInformed$
NL	: Initially the user is not informed about a malfunction of this actuator.
References	: CustomerReviewResults

Property I_MA_2

Formal	: $\neg FMInformed$
NL	: Initially the facility manager is not informed about a malfunction of this actuator.
References	: CustomerReviewResults

End of Initialization

Property P_MA_1

Formal	: $\Box([userInformed] \to \Diamond_{\leq T_{info}} \neg userInformed)$
NL	: Whenever a user is informed the predicate userInformed becomes false again within T_{info} time units.
References	: CustomerReviewResults
Comment	: By this requirement it is expressed, that the user is not permanently informed, but that is a kind of signal, e.g. an email.

Property P_MA_2

Formal	: $\Box([FMInformed] \to \Diamond_{\leq T_{info}} \neg FMInformed)$
NL	: Whenever the facility manager is informed the predicate FMInformed becomes false again within T_{info} time units.
References	: CustomerReviewResults
Comment	: By this requirement it is expressed, that the facility manager is not permanently informed, but that is a kind of signal, e.g. an email.

Property P_MA_3

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

Property P_MA_4

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

Property P_MA_5

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

Property P_MA_6

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

Description Class Actuator

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

Signature

Object *noMalAct* : *NoMalfunctionActuator*

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

Object *malAct* : *MalfunctionActuator*

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

Timed Predicate *stateManualEnteredMal*

Intention	:	This predicate is true iff it has been manually entered whether there is a malfunction of the actuator or whether there is no malfunction of it.
Scope	:	machine controlled, not visible for the environment
References	:	CustomerReviewResults, PD/FM11, PD/FMH13

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

Intention	:	This object represents a value entered manually by a user concerning a malfunction of the actuator. 1 stands for a malfunction, 0 stands for no malfunction.
Scope	:	environment controlled, not visible for the machine
References	:	PD/FM11, PD/FMH13

Timed Predicate *automaticDetMal*

Intention	:	This predicate is true iff a malfunction of the actuator has been detected automatically, i.e. not by entering it manually.
Scope	:	machine controlled, not visible for the environment
Comment	:	The specification of this function for an object of this class has to be provided by the domain expert.

Requirements/Specifications

Initialization

Property I_A_1

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

End of Initialization

Property P_A_1

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

NL : The machine assumes a malfunction of the actuator iff one of the following conditions is fulfilled:

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

References : CustomerReviewResults

Description Class RangeActuator

Intention: Class providing all definitions to describe a *range actuator*.

Comment: Note that this class can be used to describe an actuator that already exists in the environment. Therefore, the property *D_RA₁* given in this class is characterized as domain knowledge (DomKnow). Otherwise, if the actuator does not exist, it has to be a requirement, i.e. characterized as a requirement/specification (ReqSpec).

Base Classes

Class *Actuator*

Intention	:	All definitions of Actuator are also provided here.

Domain Knowledge

Property D_RA_1

Formal	:	$\Box(noMalAct.adjustedEntity \rhd^{noMalAct.modifyReaction}_{\leq noMalAct.reactionTime} noMalAct.envEntity)$
NL	:	The value of the entity 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*.
References	:	BD/Actuators: introduction

Description Class Dimmer

Intention: This class provides all definitions to describe the *dimmer* of a dimmable light.

Base Classes

Class $RangeActuator(ENVDOMAIN = PERCENT,$
$ADJUSTED\_DOMAIN = PERCENT)$

Intention	: All definitions of RangeActuator are also provided here using the values 0 to 100 (percent) as the possible values of the entity 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_DIM_1

Formal	: $\Box(noMalAct.reactionTime = 10)$
NL	: The reaction time of a signal *dimmer* is 10 milliseconds.
References	: BD/Actuators: dimmable light (dll)

Description Class BinaryActuator

Intention: This class provides all definitions to describe a binary actuator. A binary actuator is an actuator that can adjust the values 0 and 1.

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

Base Classes

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

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

Domain Knowledge

Property D_BA_1

Formal	: $\Box(\forall b \in BINARY : noMalAct.modifyReaction(b) = b)$
NL	: For a binary actuator the function representing the way in which a value adjusted by the machine (*adjustedEntity*) is modified during the reaction of the actuator, resulting in a value of the actuator in the real world (*envEntity*), is the identity.
References	: CustomerReviewResults

Description Class ControlSystemActive

Intention: This class provides all definitions to describe the signal *control system is active.*

Base Classes

Class *BinaryActuator*

Intention	: All definitions of BinaryActuator are also provided here.

Domain Knowledge

Property D_CSA_1

Formal	: $\Box(noMalAct.reactionTime = 10)$
NL	: The reaction time of a signal *control system is active* is 10 ms.
References	: BD/Actuators: dimmable light (cia)

Description Class ToggleActuator

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

Comment: 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 given in this class are characterized as domain knowledge (DomKnow). Otherwise, if the actuator does not exist, they have to be requirements, i.e. characterized as a requirement/specification (ReqSpec).

Base Classes

Class $Actuator(ENVDOMAIN = BINARY, ADJUSTED\_DOMAIN = BINARY)$

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

Domain Knowledge

Property D_TA_1

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.
References	: BD/Actuators: above the figure, CustomerReviewResults

Property D_TA_2

Formal	: $\Box(\neg noMalAct.envEntity \wedge [noMalAct.adjustedEntity] \rightarrow \Diamond_{\leq noMalAct.reactionTime} noMalAct.envEntity)$
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.
References	: BD/Actuators: above the figure, CustomerReviewResults

Description Class Pulse

Intention: This class provides all definitions to describe the *pulse line* of a dimmable light.

Base Classes

Class *ToggleActuator*

Intention	: All definitions of ToggleActuator are also provided here.

Domain Knowledge

Property D_PUL_1

Formal	: $\Box(noMalAct.reactionTime = 10)$
NL	: The reaction time of a pulse line of a dimmable light is 10 milliseconds.
References	: BD/Actuators: relais, CustomerReviewResults

Description Class ThreePointActuator

Intention: This class provides all definitions to describe a *three point actuator*. A three point actuator is an actuator which can adjust three control states: *forward*, *backward*, *stop*.

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

Base Classes

Class $RangeActuator(ENV\_DOMAIN = THREE\_PT\_ACT\_RANGE,$
$ADJUSTED\_DOMAIN = THREE\_PT\_ACT\_RANGE)$

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

Domain Knowledge

Property D_TPA_1

Formal	:	$\Box(\forall s \in THREE\_PT\_ACT\_RANGE : noMalAct.modifyReaction(s) = s)$
NL	:	For a three point actuator the function representing the way in which a value adjusted by the machine (*adjustedEntity*) is modified during the reaction of the actuator, resulting in a value of the actuator in the real world (*envStatus*), is the identity.
References	:	CustomerReviewResults

Description Class RadiatorValve

Intention: This class provides all definitions to describe a radiator valve.

Base Classes

Class *ThreePointActuator*

Intention	: All definitions of ThreePointActuator are also provided here.

Domain Knowledge

Property D_RV_1

Formal	: $\Box(noMalAct.reactionTime = 60000)$
NL	: The reaction time of a radiator valve is 60 seconds.
References	: CustomerReviewResults
Comment	: Note that the time unit is milliseconds.

Description Class SunBlind

Intention: This class provides all definitions to describe a *sun blind*. A sun blind combines a *sun blind lifter* and a *blind rotator*. In addition to the base class of a three point actuator the class SunBlind also contains two sensors recognizing whether the upper or lower end has been reached by the sun blind.

An actuator of the class SunBlind can move up and down the blind (*blind lifter* as well can rotate the blind blades (*blind rotator*). To model this behavior several states are introduced.

Base Classes

Class *ThreePointActuator*

Intention	: All definitions of ThreePointActuator are also provided here.

Signature

Object *upperEndContact* : *BinarySensor*

Intention	: This object represents the end contact detecting whether the blind is lifted completely up or not.
Scope	: environment controlled, not visible for the machine
References	: BD/Actuators: introduction, CustomerReviewResults

Object *lowerEndContact* : *BinarySensor*

Intention	: This object represents the end contact detecting whether the blind is lifted completely down or not.
Scope	: environment controlled, not visible for the machine
References	: BD/Actuators: introduction, CustomerReviewResults

Timed Predicate *moveUp*

Intention	: This predicate is true iff the blind is moving up.
Scope	: environment controlled, not visible for the machine
References	: BD/Actuators: blind lifter/rotator

Timed Predicate *moveDown*

Intention	: This predicate is true iff the blind is moving down.
Scope	: environment controlled, not visible for the machine
References	: BD/Actuators: blind lifter/rotator

Timed Predicate *moveStop*

Intention	: This predicate is true iff the blind stops, i.e. it is neither moving down nor up.
Scope	: environment controlled, not visible for the machine
References	: BD/Actuators: blind lifter/rotator

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

Intention	: This domain contains the possible values of the blind rotator, namely 0 to 180 degrees.
References	: BD/Actuators: blind lifter/rotator

Timed Function $rotateStatus() \rightarrow DEGREE$

Intention	: This function represents the current state of the blind rotator, i.e. the current position of the blind blades.
Scope	: environment controlled, not visible for the machine
References	: BD/Actuators: blind lifter/rotator

Timed Predicate $rotateIncrease$

Intention	: This predicate is true iff the blind blades are rotated in the direction of 0 degree.
Scope	: environment controlled, not visible for the machine
References	: BD/Actuators: blind lifter/rotator

Timed Predicate $rotateDecrease$

Intention	: This predicate is true iff the blind blades are rotated in the direction of 180 degree.
Scope	: environment controlled, not visible for the machine
References	: BD/Actuators: blind lifter/rotator

Timed Predicate $rotateStop$

Intention	: This predicate is true iff the blind blades do not rotate in any direction.
References	: BD/Actuators: blind lifter/rotator
Comment	: Note that this does not implies that the current rotating angle of the blades is 0 or 180 degrees.

Time Constant T_{mal}

Intention	: This time constant is used to detect automatically a malfunction of a sun blind. See property P_SBL_1 for details.
References	: CustomerReviewResults

Domain Knowledge

Property D_SBL_1

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

Property D_SBL_2

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

Property D_SBL_3

Formal	: $\Box(moveDown \rightarrow rotateStatus = 0)$
NL	: If the blinds are moved down the rotating state, i.e. the angle of the blades has to be 0 degree.
References	: BD/Actuators: blind lifter/rotator

Property D_SBL_4

Formal	: $\Box(moveUp \rightarrow rotateStatus = 180)$
NL	: If the blinds are moved up the rotating state, i.e. the angle of the blades has to be 180 degrees.
References	: BD/Actuators: blind lifter/rotator

Property D_SBL_5

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

Property D_SBL_6

Formal	: $\Box(((noMalAct.envEntity = forward) \wedge (rotateStatus = 180) \wedge \neg upperEndContact.noMalSens.envEntity) \rightarrow (moveUp \wedge rotateStop))$
NL	: Whenever the status of the sun blind is set to forward and the rotating state of the blades, i.e. the angle, is 180 degrees and the blind is not at the upper end, the blind is lifted up and the blades are not rotating.
References	: BD/Actuators: blind lifter/rotator

Property D_SBL_7

Formal	: $\Box(((noMalAct.envEntity = backward) \wedge (rotateStatus = 0) \wedge \neg lowerEndContact.noMalSens.envEntity) \rightarrow (moveDown \wedge rotateStop))$
NL	: Whenever the status of the sun blind is set to backward and the rotating state of the blades, i.e. the angle, is 0 degrees and the blind is not at the lower end, the blind is lifted down and the blades are not rotating.
References	: BD/Actuators: blind lifter/rotator

Property D_SBL_8

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

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

References : BD/Actuators: blind lifter/rotator

Property D_SBL_9

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

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

References : BD/Actuators: blind lifter/rotator

Property D_SBL_{10}

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

References : BD/Actuators: blind lifter/rotator

Comment : This has to be made more precise by the domain expert.

Requirements/Specifications

Property P_SBL_1

Formal : $\Box(\blacksquare_{\geq T_{mal}}(noMalAct.adjustedEntity = forward \wedge$
$lowerEndContact.noMalSens.convertedEntity)$
$\rightarrow automaticDetMal)$

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

References : CustomerReviewResults

Property P_SBL_2

Formal : $\Box(\blacksquare_{\geq T_{mal}}(noMalAct.adjustedEntity = backward \wedge$
$upperEndContact.noMalSens.convertedEntity)$
$\rightarrow automaticDetMal)$

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

References : CustomerReviewResults

Description Class DimmableLight

Intention: This class provides all definitions concerning a dimmable light. These are w.r.t. the BD document: pulse, status line, control system is active, dim value.

Base Classes

Class *Light*

Intention	:	All definitions of Light are also provided here.
References	:	BD/Actuators: figure, BD/Actuators: above the figure, BD/Sensors: dimmable light, BD/Offices

Signature

Object *pulse* : *Pulse*

Intention	:	This object represents the pulse line of a dimmable light.
Scope	:	environment controlled, not visible for the machine
References	:	BD/Actuators: figure, BD/Actuators: above the figure, BD/Sensors: relais, CustomerReviewResults

Object *conSysActive* : *ControlSystemActive*

Intention	:	This object represents the signal *control system is active* of a dimmable light
Scope	:	environment controlled, not visible for the machine
References	:	BD/Actuators: figure, BD/Actuators: above the figure, BD/Sensors: dimmable light (cia)

Object *dimmer* : *Dimmer*

Intention	:	This object represents a dimmer actuator.
Scope	:	environment controlled, not visible for the machine
References	:	BD/Actuators: figure, BD/Actuators: above the figure, BD/Sensors: dimmable light (dll), BD/Offices

Timed Function $lightValue() \rightarrow PERCENT$

Intention	:	This function represents the environmental light value (given in percent), resulting from a certain dimmer adjustment.
Scope	:	environment controlled, not visible for the machine
Comment	:	Note, that there is a difference between lightValue and dimmer.noMalAct.envEntity: dimmer.noMalAct.envEntity refers to the environmental value set by the dimmer actuator, e.g. a voltage; lightValue refers to a real light value, as it would result from a corresponding dimmer setting.

Time Constant *safeModeReactDelay*

Intention	: This time constant represents the delay between a missing control-system-is-active signal and the appropriate reaction, i.e. changing to fail safe mode.
Scope	: machine controlled, not visible for the environment
References	: BD/Actuators: above the figure
Comment	: This entity is explicitly introduced by the customer.

Timed Predicate *safeMode*

Intention	: This predicate is true iff the dimmable light is in fail safe mode.
Scope	: machine controlled, not visible for the environment
References	: BD/Actuators: above the figure
Comment	: This entity is explicitly introduced by the customer.

Domain Knowledge

Property D_DL_1

Formal	: $\Box(\blacksquare_{>60000}\neg conSysActive.noMalAct.envEntity \Rightarrow_{\leq safeModeReactTime} safeMode)$
NL	: Whenever there was no control-system-is-active signal for at least 60 seconds, the dimmable light is in safe mode within safe mode delay at least as long as the precondition holds.
References	: BD/Actuators: above the figure
Comment	: Note that the time unit is milliseconds.

Property D_DL_2

Formal	: $\Box(safeMode \rightarrow dimmer.noMalAct.envEntity = 100)$
NL	: Whenever the dimmable light is in safe mode the dimmer value is 100 percent.
References	: BD/Actuators: above the figure

Property D_DL_3

Formal	: $\Box(noMalSens.envEntity \leftrightarrow lightValue \neq 0)$
NL	: The light is off (on), iff the percental light value is zero (not zero).
References	: CustomerReviewResults

Description Class LightButtonSet

Intention: This class provides all definitions concerning the combination of a number of push buttons with a light. The number of push buttons is not relevant for the behaviour of the control system.

Signature

Object *light* : *Light*

Intention	: This object represents the light that is controlled by the push buttons.
Scope	: environment controlled, not visible for the machine
References	: BD/Rooms, BD/Offices, BD/Hallway

Timed Predicate *pushButton*

Intention	: This predicate is true iff any button is pushed.
Scope	: environment controlled, not visible for the machine
References	: BD/Rooms, BD/Offices, BD/Hallway

Time Constant T_LBS_1

Intention	: This time constant represents the delay between a button push and the change of the light. Here the light existing in the real world is considered.

Time Constant T_LBS_2

Intention	: This time constant represents the delay between a button push and the change of the status line of the light. Here the signal sent to the machine and not the light existing in the real world is considered.

Domain Knowledge

Property D_LBS_1

Formal	: $\Box(T\_LBS_2 > T\_LBS_1)$
NL	: The time constant T_LBS_2 is always greater than the time constant T_LBS_1.
References	: CustomerReviewResults

Property D_LBS_2

Formal	: $\Box(light.noMalSens.envEntity = 0 \wedge [pushBut] \rightarrow \Diamond_{\leq T\_LBS_1} light.noMalSens.envEntity = 1)$
NL	: Whenever the light is off and any push button is pressed, the light changes to on within time T_LBS_1.
References	: BD/Rooms, BD/Offices, BD/Hallway
Comment	: Note that this property expresses domain knowledge which is not visible for the machine. The corresponding property which is visible for the machine is D_LBS_4.

Property D_LBS_3

Formal	: $\Box(light.noMalSens.envEntity = 1 \wedge [pushBut] \rightarrow \Diamond_{\leq T\_LBS_1} light.noMalSens.envEntity = 0)$
NL	: Whenever the light is on and any push button is pressed, the light changes to off within time T_LBS_1.
References	: BD/Rooms, BD/Offices, BD/Hallway
Comment	: Note that this property expresses domain knowledge which is not visible for the machine. The corresponding property which is visible for the machine is D_LBS_5.

Property D_LBS_4

Formal	: $\Box(light.noMalSens.convertedEntity = 0 \wedge [pushBut] \rightarrow \Diamond_{\leq T\_LBS_2} light.noMalSens.convertedEntity = 1)$
NL	: Whenever the light status line is 0 and any push button is pressed, the status line changes to 1 within time T_LBS_1.
References	: BD/Rooms, BD/Offices, BD/Hallway

Property D_LBS_5

Formal	: $\Box(light.noMalSens.convertedEntity = 1 \wedge [pushBut] \rightarrow \Diamond_{\leq T\_LBS_2} light.noMalSens.convertedEntity = 0)$
NL	: Whenever the light status line is 0 and any push button is pressed, the status line changes to 1 within time T_LBS_1.
References	: BD/Rooms, BD/Offices, BD/Hallway

Description Class DimmableLightButtonSet

Intention: This class provides all definitions to extend the LightButtonSet for dimmable lights.

Base Classes

Class *LightButtonSet*

Intention	: All definitions of LightButtonSet are also provided here.

Signature

Object $LightButtonSet :: light : DimmableLight$

Intention	: This object represents the dimmable light as a special case of the light in the light-button set.
Scope	: environment controlled, not visible for the machine
References	: BD/Rooms, BD/Actuators: above the figure

Domain Knowledge

Property D_DLBS_1

Formal	: $\Box(\forall d \in PERCENT : light.noMalSens.convertedEntity = 0 \wedge [pushBut] \wedge dimmer.noMalAct.adjusted = d \rightarrow \Diamond_{\leq T\_LBS_1} dimmer.noMalAct.envEntity = d)$
NL	: Whenever the light status line is off and any push button is pushed and the dimmer is set to d percent, the light is dimmed within the push-to-light delay to the value given by the dimmer, i.e. to d.
References	: BD/Rooms, BD/Actuators: above the figure

Property D_DLBS_2

Formal	: $\Box(light.noMalSens.convertedEntity = 1 \wedge [pushBut] \rightarrow \Diamond_{\leq T\_LBS_1} light.noMalSens.envEntity = 0)$
NL	: Whenever the light status line is on and any push button is pushed, the ceiling light is off within the push-to-light delay.
References	: BD/Rooms, BD/Actuators: above the figure

Description Class Radiator

Intention: This class provides all the definitions belonging to a radiator.

Signature

Object *waterInTemp* : *WaterTempSensor*

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

Object *waterOutTemp* : *WaterTempSensor*

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

Object *meanTemp* : *WaterTempSensor*

Intention	: This object represents a sensor to measure the mean radiator temperature of a radiator.
References	: BD/Rooms

Object *valveAct* : *RadiatorValve*

Intention	: This object represents the radiator control valve of a radiator.
References	: BD/Rooms

Description Class WindowSection

Intention: This class provides all the definitions belonging to each window section. This is a radiator. This is the usual window section of rooms that are situated on the northern site of a building.

Signature

Object *rad* : *Radiator*

Intention	: This object represents the radiator of a window section.
References	: BD/Rooms

Description Class BlindedWindowSection

Intention: This class provides all definitions to describe a *blinded window section*. In addition to a *normal* window section this class provides also a *sun blind* including a *sun blind lifter* and a *blind rotator*. This is the usual window sections of rooms that are not situated on the northern site of a building.

Base Classes

Class *WindowSection*

Intention	: All definitions of WindowSection are also provided here.

Signature

Object *sb* : *SunBlind*

Intention	: This object represents a sun blind.
References	: BD/Rooms

Description Class Door

Intention: This class provides all definitions to describe a door with it's various contacts.

Signature

Object *doorClosedContact* : *DoorContact*

Intention	: This object represents the closed contact of a door. It is true iff the door is completely closed.
Scope	: environment controlled, not visible for the machine
References	: BD/Sensors: door closed contact (dcc)

Object *doorOpenContact* : *DoorContact*

Intention	: This object represents the open contact of a door. It is true iff the door is open for at least 30 degree.
Scope	: environment controlled, not visible for the machine
References	: BD/Sensors: door open contact (doc)

Object *doorLockContact* : *DoorContact*

Intention	: This object represents the lock contact of a door. It is true iff the door is locked by a key.
Scope	: environment controlled, not visible for the machine
References	: BD/Sensors: door lock contact (dlc)

Description Class LightScene

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

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

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

Signature

Domain $LAMPS > \{WINDOW, WALL\}$

Intention : This domain contains the installed lightings that can be used to establish a certain light scene.

- *WINDOW* stands for the window ceiling light and
- *WALL* for the wall ceiling light.

References : DIC/light scene

Timed Function $name() \rightarrow STRING$

Intention : This function represents the name of the light scene.
References : DIC/light scene

Timed Function $ambientIlluminance() \rightarrow LIGHT\_VALUES$

Intention : This function represents the desired ambient light intensity of a room.
References : DIC/light scene

Timed Function $firstAmbientLighting() \rightarrow LAMPS$

Intention : This function represents the first lighting that should be used to reach the desired ambient light of a room.
References : DIC/light scene

Timed Function $secondAmbientLighting() \rightarrow LAMPS$

Intention : This function represents the second lamp that should be used to reach the desired ambient light of a room.
References : DIC/light scene

Requirements/Specifications

Property P_LS_1

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

Description Class LightSceneOffice

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

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

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

Signature

Timed Function $\mathit{firstDeskIlluminaire}() \rightarrow \mathit{LAMPS}$

Intention	:	This function represents the first lamp that should be used to reach the desired desk light of a room.
References	:	DIC/light scene

Timed Function $\mathit{secondDeskIlluminaire}() \rightarrow \mathit{LAMPS}$

Intention	:	This function represents the second lamp that should be used to reach the desired desk light of a room.
References	:	DIC/light scene

Requirements/Specifications

Property P_LSO_1

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

Description Class OccDelayTimes

Intention: This class provides all definitions to describe the values of the three time delays dt_1, dt_2, dt_3, used to determine occupancy of an area (see description class *NoMalfunctionOccupancy*). The values of the three time delays depend on the inputs given by the user and the facility manager w.r.t. this time delays. These time delays can be set for different expected *usage profile* of an area (see description class *TemperatureCtrlPanel*).

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

Signature

Object dt_1 : $ControlledInput(INPUT\_DOM = DT1\_RANGE)$

Intention	: This object represents the current value of the time delay dt_1 (depending on the values entered by the facility manager and the room user).
Scope	: environment controlled, not visible for the machine
References	: PD/Paragraph 7
Comment	: This object is specific for each room type.

Object dt_2 : $ControlledInput(INPUT\_DOM = DT2\_RANGE)$

Intention	: This object represents the current value of the time delay dt_2 (depending on the values entered by the facility manager and the room user).
Scope	: environment controlled, not visible for the machine
References	: PD/Paragraph 7
Comment	: This object is specific for each room type.

Object dt_3 : $ControlledInput(INPUT\_DOM = DT3\_RANGE)$

Intention	: This object represents the current value of the time delay dt_3 (depending on the values entered by the facility manager and the room user).
Scope	: environment controlled, not visible for the machine
References	: PD/Paragraph 7
Comment	: This object is specific for each room type.

Requirements/Specifications

Property P_ODT_1

Formal	: $dt_1.default.checked = 60000$
NL	: The initial default value of time dt_1 given by the facility manager is 1 minute.
References	: PD/Paragraph 7, CustomerReviewResults
Comment	: Note that only an initial value is specified by this property. Note that the time unit is milliseconds.

Property P_ODT_2

Formal	: $dt_2.default.checked = 600000$
NL	: The initial default value of time dt_2 given by the facility manager is 10 minutes.
References	: PD/Paragraph 7, CustomerReviewResults
Comment	: Note that only an initial value is specified by this property. Note that the time unit is milliseconds.

Property P_ODT_3

Formal	: $dt_3.default.checked = 1800000$
NL	: The initial default value of time dt_3 given by the facility manager is 30 minutes.
References	: PD/Paragraph 7, CustomerReviewResults
Comment	: Note that only an initial value is specified by this property. Note that the time unit is milliseconds.

Description Class LightCtrlPanelFM

Intention: This class provides all the definitions of entities concerning the control of the light in a room the facility manager can enter using the control panel. Here we consider the values given for *one single* room, not for all or a set of rooms.

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

Signature

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

Intention	: This object represents the time an area has to be unoccupied continuously before the lights in this area have to be switched off.
References	: PD/FM4, PD/FM5

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

Intention	: This object represents whether the facility manager has turned all lights on or off in a certain area. *lightOff.checked* = 1 means that the lights have to be on; *lightOff.checked* = 0 means that the lights have to be off.
Scope	: environment controlled, not visible for the machine
References	: PD/FM6, CustomerReviewResults

Description Class LightCtrlPanelRoom

Intention: This class provides all the definitions of entities concerning the control of the light in a room a user can enter using the control panel and for which the facility manager can not enter a default value. Values for which the facility manager specifies a default value that can be changed by a user are considered in the description class LightCtrlPanel.

This control panel includes the settings for the sun blinds in a room a user can make.

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

Formal Parameters

Function $numSunBlinds() \rightarrow NAT$

Intention	:	This function represents the number of sun blinds that belong to a room and which can be controlled using the control panel.

Signature

Object $chosenLightScene$: $EnteredValue(CHECKED\_DOM = STRING)$

Intention	:	This object represents the light scene currently chosen by using the control panel.
Scope	:	environment controlled, not visible for the machine
References	:	PD/U6

Object $ambLightLev$: $EnteredValue(CHECKED\_DOM = LIGHT\_VALUES)$

Intention	:	This object represents the value defined for the ambient light level by using the control panel.
Scope	:	environment controlled, not visible for the machine
References	:	PD/U14, PD/U16, PD/U7
Comment	:	Note that this setting is only relevant if the light is not controlled manually using the push buttons.

Object $winCeilLightSwitch$: $EnteredValue(CHECKED\_DOM = BINARY)$

Intention	:	This object represents the value defined for the window ceiling light by using the control panel. $winCeilLightSwitch.checked = 1$ means that the window ceiling light should be on. $winCeilLightSwitch.checked = 0$ means that the window ceiling light should be off.
Scope	:	environment controlled, not visible for the machine
References	:	PD/U14, PD/U16, CustomerReviewResults
Comment	:	Note that this setting is only relevant if the light is not controlled manually using the push buttons.

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

Intention	: This object represents the value defined for the wall ceiling light by using the control panel. *wallCeilLightSwitch.checked* = 1 means that the wall ceiling light should be on. *wallCeilLightSwitch.checked* = 0 means that the wall ceiling light should be off.
Scope	: environment controlled, not visible for the machine
References	: PD/U14, PD/U16, CustomerReviewResults
Comment	: Note that this setting is only relevant if the light is not controlled manually using the push buttons.

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

Intention	: This object represents the value that indicates whether the currently set ambient light level (*ambLightLev*) should be applied (this is the case for *ambientSwitch.checked* = 1) or the currently settings of the window and wall ceiling light switches of the control panel should be applied (this is the case for *ambientSwitch.checked* = 0).
Scope	: environment controlled, not visible for the machine
References	: CustomerReviewResults
Comment	: Note that this setting is only relevant if the light is not controlled manually using the push buttons.

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

Intention	: This object represents the value that indicates whether the light scenes are enabled by the control panel, i.e. the currently chosen light scene should be applied (this is the case for *lightSceneSwitch* = 1) or the currently settings of the control panel concerning the ambient switch (*ambientSwitch*), the ceiling light switches (*winCeilLightSwitch*,*winCeilLightSwitch*), and the ambient light level (*ambLightLev*) should be applied (this is the case for *lightSceneSwitch* = 0).
Scope	: environment controlled, not visible for the machine
References	: CustomerReviewResults
Comment	: Note that this setting is only relevant if the light is not controlled manually using the push buttons.

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

Intention	: Each object represents the value set by the user for the control of a sun blind by using the control panel.
Scope	: environment controlled, not visible for the machine
References	: PD/UH6, CustomerReviewResults

Requirements/Specifications

Property P_LCP_1

NonFormal	:	The control panel should be easy and intuitive to use.
References	:	PD/NF9

Property P_LCP_2

NonFormal	:	Light scenes can be entered using the control panel which are stored and can be chosen.
References	:	PD/U6
Comment	:	Note that this is not the same as the currently chosen light scene.

Description Class LightCtrlPanel

Intention: This class combines the two control classes LightCtrlPanelFM and LightCtrlPanelRoom resulting in a class that contains all entities concerning the control of the lights in a room that can be entered using a control panel. This combination is just a collection of the entities of the two mentioned classes.

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

Signature

Object *lCPFM* : *LightCtrlPanelFM*

Intention : This object represents the part of the light control panel containing the entities that can be entered by the facility manager.

Object *lCPRoom* : *LightCtrlPanelRoom*

Intention : This object represents the part of the light control panel containing the entities that can be entered by a user in a room.

Description Class TemperatureCtrlPanelFM

Intention: This class provides all the definitions of values concerning the control of the temperature in a room the facility manager can enter using the control panel and which can not be changed by a user entering a value at the control panel in a room. Values for which the facility manager specifies a default value that can be changed by a user are considered in the description class TemperatureCtrlPanel.

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

Signature

Object *heatingPeriod* : *CalendarInterval*

Intention	: This object represents the heating period, i.e. the start and end point of the period and whether the actual date is within the non heating period or not.
Scope	: environment controlled, not visible for the machine
References	: PD/FMH4
Comment	: This object is the same for all rooms.

Object *nonHeatingPeriod* : *CalendarInterval*

Intention	: This object represents the non heating period, i.e. the start and end point of the period and whether the actual date is within the non heating period or not.
Scope	: environment controlled, not visible for the machine
References	: PD/FMH4
Comment	: This object is the same for all rooms.

Object *buildingNonFreezePeriod* : *CalendarInterval*

Intention	: This object represents the period within which the non freeze mode is set for the whole building.
Scope	: environment controlled, not visible for the machine
References	: PD/FMH5, CustomerReviewResults
Comment	: This object is the same for all rooms.

Object *nonFreezePeriod* : *CalendarInterval*

Intention	: This object represents the period within which the non freeze mode is set for a certain room.
Scope	: environment controlled, not visible for the machine
References	: PD/FMH5, CustomerReviewResults
Comment	: This object is specific for each room.

Timed Predicate *roomNonFreezeDefined*

Intention	: This predicate is true iff the facility manager has defined a non-freeze period for a certain room.
References	: PD/FMH5
Comment	: This object is specific for each room.

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

Intention	: This object represents the temperature that should be applied if the operation mode *NonFreezeMode* is active.
Scope	: environment controlled, not visible for the machine
References	: PD/FMH5
Comment	: This object is the same for all rooms.

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

Intention	: This object represents the temperature that should be applied if the operation mode *standbyMode* is active. This temperature is set by the facility manager.
Scope	: environment controlled, not visible for the machine
References	: CustomerReviewResults
Comment	: This object is specific for each room type.

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

Intention	: This object represents the temperature that should be applied if the operation mode *OffMode* is active. This temperature is set by the facility manager.
Scope	: environment controlled, not visible for the machine
References	: CustomerReviewResults
Comment	: This object is specific for each room type.

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

Intention	: This object represents the modus set by the facility manager indicating whether the delay times used for determining occupancy are adaptive (i.e. the systems *learns* a typical user behavior) or not.
Scope	: environment controlled, not visible for the machine
References	: PD/Paragraph 7, PD/FMH6
Comment	: This object is the same for all rooms.

Requirements/Specifications

Initialization

Property P_TCPF_1

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

End of Initialization

Description Class TemperatureCtrlPanelRoom

Intention: This class provides all the definitions of values concerning the control of the temperature in a room a user can enter using the control panel and for which the facility manager can not enter a default value. Values for which the facility manager specifies a default value that can be changed by a user are considered in the description class TemperatureCtrlPanel.

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

Signature

Object *nonFreezeOverride* : *EnteredValue(BINARY)*

Intention	: This object represents whether the user has entered an override over the non-freeze operation mode set by the facility manager or not.
References	: PD/UH3

Description Class TemperatureCtrlPanel

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

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

Base Classes

Object *tCPFM* : *TemperatureCtrlPanelFM*

Intention	: This object represents the part of the light control panel containing the entities that can be entered by the facility manager.

Object *tCPRoom* : *TemperatureCtrlPanelRoom*

Intention	: This object represents the part of the light control panel containing the entities that can be entered by a user in a room.

Signature

Object *expectedUsageTimes* : *OccDelayTimes*

Intention	: This object represents the set of occupancy delay times dt_1, dt_2, dt_3 in the case that a usage of the room is expected.
Scope	: environment controlled, not visible for the machine
References	: PD/Paragraph 7
Comment	: This object is specific for each room.

Object *notExpectedUsageTimes* : *OccDelayTimes*

Intention	: This object represents the set of occupancy delay times dt_1, dt_2, dt_3 in the case that a usage of the room is not expected.
Scope	: environment controlled, not visible for the machine
References	: PD/Paragraph 7
Comment	: This object is specific for each room.

Object *comfortTemperature* : *ControlledInput*$(INPUT\_DOM = COMFORT\_TEMP)$

Intention	: This object represents the current comfort temperature of a room (depending on the values entered by the facility manager and the room user).
Scope	: environment controlled, not visible for the machine
References	: PD/UH1, PD/FMH3
Comment	: The default value set by the facility manager is specific for each room type.

Requirements/Specifications

Property P_TCP_1

NonFormal	: The control panel should be easy and intuitive to use.
References	: PD/NF9

Description Class ExpectedUsage

Intention: This class provides all the definitions to describe the *expected usage* of a room. This depends on several settings of the facility manager and the user.

Signature

Object $workingTime$:
$ControlledInput(INPUT\_DOM = WORKING\_TIME\_RANGE)$

Intention	:	This object represents the current working time (depending on the values entered by the facility manager and the room user).
Scope	:	environment controlled, not visible for the machine
References	:	PD/FMH3, PD/UH4
Comment	:	This object is specific for each room type.

Object $holidayTime$: $ControlledInput(INPUT\_DOM = HOLIDAY\_TIME\_RANGE)$

Intention	:	This object represents the current holiday time (depending on the values entered by the facility manager and the room user).
Scope	:	environment controlled, not visible for the machine
References	:	PD/FMH3, PD/UH5
Comment	:	This object is specific for each room type.

Timed Predicate $expectedUsage$

Intention	:	This predicate is true iff a usage of a room is expected by the machine.
Scope	:	machine controlled, not visible for the environment
Comment	:	The validity of this predicate depends on the current working and holiday time settings of the room user and the facility manager.

Requirements/Specifications

Property D_EU_1

Formal : $\Box(expectedUsage \leftrightarrow now \in workingTime.value \land now \notin holidayTime.value)$

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

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

References : CustomerReviewResults

Description Class AdaptiveOccupancyTimes

Intention: This class provides all the definitions to describe how the three time delays used to determine occupancy in an area are modified if the facility manager has chosen the delay times to be *adaptive*.

Comment: Diese Klasse ist noch nicht vollständig modelliert, weil noch einige Unklarheiten existieren.

Signature

Timed Function $dt_1Adaptive() \rightarrow TIME$

Intention	: This function represents the value for dt_1 as it is determined using an adaptive learning function.
Scope	: machine controlled, not visible for the environment
References	: PD/FMH6, PD/Paragraph 7

Time Constant T_{pastDT_1}

Intention	: interval in the past considered for computing the value of time delay dt1
Scope	: machine controlled, visible for the machine
References	: PD/Paragraph 7

Timed Function $dt_2Adaptive() \rightarrow TIME$

Intention	: This function represents the value for dt_2 as it is determined using an adaptive learning function.
Scope	: machine controlled, not visible for the environment
References	: PD/FMH6, PD/Paragraph 7

Timed Function $dt_3Adaptive() \rightarrow TIME$

Intention	: This function represents the value for dt_3 as it is determined using an adaptive learning function.
Scope	: machine controlled, not visible for the environment
References	: PD/FMH6, PD/Paragraph 7

Description Class NoMalfunctionOccupancy

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

Whether the machine assumes occupancy in an area or not depends on three time values, *dt1*, *dt2*, *dt3*, and on a predicate indicating whether a person is currently in the area or not. Note that the status of *occupancy* in general does not correspond with the fact whether there is a person in an area or not. *Occupancy* is derived term depending on this fact.

Signature

Parameter

Timed Function $dt_1() \rightarrow TIME$

Intention	: This function represents the current value used for the first time delay used to determine occupancy.
Scope	: machine controlled, not visible for the environment
References	: PD/Paragraph 7
Comment	: The precise correlation between this time delay and *occupancy* is given in the property P_NMO_1. This value has to be specified when an object of this class is used.

Timed Function $dt_2() \rightarrow TIME$

Intention	: This function represents the current value used for the second time delay used to determine occupancy.
Scope	: machine controlled, not visible for the environment
References	: PD/Paragraph 7
Comment	: The precise correlation between this time delay and *occupancy* is given in property P_NMO_1. This value has to be specified when an object of this class is used.

Timed Function $dt_3() \rightarrow TIME$

Intention	: This function represents the current value used for the third time delay used to determine occupancy.
Scope	: machine controlled, not visible for the environment
References	: PD/Paragraph 7
Comment	: The precise correlation between this time delay and *occupancy* is given in property P_NMO_1. This value has to be specified when an object of this class is used.

Timed Predicate *emptyMach*

Intention	: This predicate is true iff the machine assumes that there is currently no person in the area for which occupancy is considered.
Scope	: machine controlled, not visible for the environment
References	: PD/Paragraph 7
Comment	: This predicate has to be specified when an object of this class is used (e.g. using the values of motion detectors).

END OF PARAMETER

Timed Predicate *emptyEnv*

Intention	: This predicate is true iff there is in the real world no person in the area for which occupancy is considered.
Scope	: environment controlled, not visible for the machine
References	: PD/Paragraph 7

Timed Predicate *occupied*

Intention	: This predicate is true iff the machine assumes that the area for which occupancy is considered is occupied.
Scope	: machine controlled, not visible for the environment
References	: PD/Paragraph 4, PD/Paragraph 7
Comment	: This predicate represents the situation as it is seen by the machine and not how it is in the real world. Note that this predicate does not describe whether a person is in a area or not. See Timed Predicate *emptyMach*.

REQUIREMENTS/SPECIFICATIONS

Property P_NMO_1

Formal : $\Box(occupied \leftrightarrow$

$$\blacksquare_{\leq dt_1} \neg emptyMach \vee$$
$$< *[\blacksquare_{\leq dt_1+dt_2} \neg emptyMach] \Rightarrow> \blacksquare\blacklozenge_{\leq dt_3} \neg emptyMach)$$

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

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

References : PD/Paragraph 7

Description Class MalfunctionOccupancy

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

In the case of a malfunction of 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.

Signature

Timed Predicate *malOccDetEnv*

Intention	: This predicate is true iff some sensors existing in the real world used to determine occupancy have a malfunction.
Scope	: environment controlled, not visible for the machine
References	: PD/NF5, PD/NFH4
Comment	: Note that this predicate represents the value as it exists in the real world. This value is not visible for the machine. It can be used to express requirements.

Timed Predicate *malOccDetMach*

Intention	: This predicate is true iff the machine assumes that some sensors used to determine occupancy have a malfunction.
Scope	: machine controlled, not visible for the environment
References	: PD/NF5, PD/NFH4
Comment	: Note that this predicate represents the value as it seen by the machine.

Object $manualEnteredOcc : EnteredValue(CHECKED\_DOM = BINARY)$

Intention	: This object represents the value concerning occupancy entered manually by a user in in the case of a malfunction of some sensors used to determine occupancy.
Scope	: environment controlled, not visible for the machine
References	: PD/NFH1, PD/NFH4

Timed Predicate *stateManualEnteredOcc*

Intention	: This predicate is true iff it has been entered manually whether there is occupancy or not.
Scope	: machine controlled, not visible for the environment
References	: PD/NFH1, PD/NFH4
Comment	: This entity is explicitly introduced by the customer.

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.
Scope	: machine controlled, not visible for the environment
References	: PD/NF5, PD/NFH4
Comment	: The specification of this predicate differs for the light control and the temperature control.

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.
Scope	: machine controlled, not visible for the environment
References	: PD/NF5, PD/NFH4

Requirements/Specifications

Initialization

Property I_MO_1

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

End of Initialization

Property P_MO_1

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.

References : PD/NF5, PD/NFH4, CustomerReviewResults

Description Class Occupancy

Intention: This class provides all definitions to describe the occupancy in a specific area.

Base Classes

Object *noMalOcc* : *NoMalfunctionOccupancy*

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

Object *malOcc* : *MalfunctionOccupancy*

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

Signature

Timed Predicate *stateManualEnteredMal*

Intention	:	This predicate is true iff it has been manually entered whether there is a malfunction of the sensors determining occupancy or whether there is no malfunction of them.
Scope	:	machine controlled, not visible for the environment
References	:	CustomerReviewResults, PD/FM11, PD/FMH13

Object *manualEnteredMal* : $EnteredValue(CHECKED\_DOM = BINARY)$

Intention	:	Manually entered value concerning a malfunction of the sensors determining occupancy entered by a user. 1 stands for a malfunction, 0 stands for no malfunction.
Scope	:	environment controlled, not visible for the machine
References	:	PD/FM11, PD/FMH13

Timed Predicate *automaticDetMal*

Intention	:	This predicate is true iff a malfunction of the sensors used to determine occupancy has been detected automatically, i.e. not by entering it manually.
Scope	:	machine controlled, not visible for the environment
References	:	CustomerReviewResults
Comment	:	The specification of this function for an object of this class has to be provided by the domain expert.

Timed Predicate *occUsed*

Intention	: This predicate is true iff the machine assumes occupancy.
Scope	: machine controlled, not visible for the environment
References	: PD/NFH4, PD/U1, PD/U2, PD/U3, PD/U4, PD/FM3, PD/FM6, PD/U17, PD/NF5, PD/UH3
Comment	: This entity is explicitly introduced by the customer.

Requirements/Specifications

Initialization

Property I_OCC_1

Formal	: $\neg stateManualEnteredMal$
NL	: Initially it has not been entered manually whether there is a malfunction of the sensors determining occupancy or whether there is no malfunction of them
References	: CustomerReviewResults

End of Initialization

Property P_OCC_1

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

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

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

References : CustomerReviewResults

Property P_OCC_2

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.

References : CustomerReviewResults

Description Class TempOperationMode

Intention: This class provides all the definitions to describe a single *operation mode* of a room concerning temperature.

Signature

Timed Predicate *active*

Intention	: This predicate is true iff this operation mode is active.
Scope	: environment controlled, not visible for the machine
References	: PD/Paragraph 4

Timed Function $desiredTemperature() \rightarrow TEMP$

Intention	: This function represents the temperature that should be established in a room in this operation mode.
Scope	: machine controlled, not visible for the environment
References	: PD/FMH5, PD/UH1, CustomerReviewResults
Comment	: This entity is explicitly introduced by the customer.

Description Class TempOperationModeSet

Intention: This class provides all the definitions to describe all the *operation modes* of a room concerning temperature.

Four operation modes are distinguished: *NonFreezeMode*, *ComfortMode*, *StandbyMode*, and *OffMode*. The definitions of these modes are given in this class. Which mode is currently active depends on three predicates: *expectedUsage*, *occUsed*, and *nonFreezeTime*. The definitions of the four operation modes is well defined. This means that it is *correct* (i.e. at each time point at most one mode is active) and *complete* (i.e. at each time point at least one mode is active).

Signature

Object *expUse* : *ExpectedUsage*

Intention	:	This object represents the part of the definition of the operation modes defining whether a usage of a room is expected or not.

Object *occ* : *Occupancy*

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

Object *OffMode* : *TempOperationMode*

Intention	:	This object represents the operation mode *offMode*.
Scope	:	environment controlled, not visible for the machine
References	:	PD/Paragraph 4, PD/UH4, PD/UH5

Object *StandbyMode* : *TempOperationMode*

Intention	:	This object represents the operation mode *StandbyMode*.
Scope	:	environment controlled, not visible for the machine
References	:	PD/Paragraph 4, PD/UH4

Object *ComfortMode* : *TempOperationMode*

Intention	:	This object represents the operation mode *ComfortMode*.
Scope	:	environment controlled, not visible for the machine
References	:	PD/Paragraph 4, PD/UH1

Object *NonFreezeMode* : *TempOperationMode*

Intention	:	This object represents the operation mode *NonFreezeMode*.
Scope	:	environment controlled, not visible for the machine
References	:	PD/Paragraph 4, PD/UH3, PD/FMH5

Timed Predicate *nonFreezeTime*

Intention	:	This predicate is true iff it is currently non-freeze operation time.
Scope	:	machine controlled, not visible for the environment
References	:	PD/Paragraph 4, PD/UH3, PD/FMH5

Timed Function $roomDesiredTemperature() \rightarrow TEMP$

Intention	: This function represents the current desired temperature that should be established in a room.
Scope	: machine controlled, not visible for the environment
References	: PD/UH1, PD/FMH5
Comment	: This entity is explicitly introduced by the customer.

Requirements/Specifications

Property P_TOMS_1

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

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

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

References : PD/Paragraph 4

Property P_TOMS_2

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

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

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

References : PD/Paragraph 4

Property P_TOMS_3

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

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

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

References : PD/Paragraph 4

Property P_TOMS_4

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

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

References : PD/Paragraph 4

Property P_TOMS_5

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

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

References : CustomerReviewResults

Property P_TOMS_6

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

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

References : CustomerReviewResults

Property P_TOMS_7

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

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

References : CustomerReviewResults

Property P_TOMS_8

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

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

References : CustomerReviewResults, PD/FMH5

Description Class AreaLight

Intention: This class provides all definitions to describe general areas with light (here: rooms and hallways).

MEHR ERLÄUTERUNG!!!

Signature

Object *occ* : *Occupancy*

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

Timed Predicate: *safeLight*

Intention	: This predicate is true, iff there is sufficient light to move safely.
Scope	: environment controlled, not visible for the machine
References	: PD/U1, PD/U17

Time Constant T_AL_1

Intention	: This time constant represents the maximal time delay between the occupation of an area and the establishment of safe light.

Timed Predicate *allLightsOff*

Intention	: This predicate is true iff all lights are off in an area.
Scope	: environment controlled, not visible for the machine
References	: PD/FM6
Comment	: Note, that by this predicate the situation existing in the real world is considered.

Timed Predicate *noSaveLight*

Intention	: This predicate is true iff the light intensity in an area that is occupied can be less than the light intensity needed to establish *save light*.
Scope	: machine controlled, not visible for the environment
References	: PD/U1

Requirements/Specifications

Initialization

Property I_AL_1

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

End of Initialization

Property P_AL_1

Formal	: $\Box(\neg noSaveLight \wedge occ.occUsed \Rightarrow_{\leq T\_AL_1} safeLight)$
NL	: Whenever save 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.
References	: PD/U1, PD/U17

Property P_AL_2

Formal	: $\Box(occ.automaticDetermValue)$
NL	: The value determined automatically by the machine for occupancy concerning the light control in the case of a malfunction of some sensors used to determine occupancy concerning the light control is always true.
References	: PD/NF5

Description Class RoomLight

Intention: This class provides all definitions to describe the light environment and requirements common to any kind of room. The specified behaviour of the lights considers the case of an occupied room. Together with the behaviour of the base class AreaLight for non-occupied rooms the specification is well-defined and complete.

Base Classes

Class *AreaLight*

Intention	:	All definitions of Area are also provided here.

Signature

Object cp : *LightCtrlPanel*

Intention	:	This object represents the light specific inputs of a user and facility manager.
References	:	BD/Rooms

Object $win, wall$: *DimmableLightButtonSet*

Intention	:	This objects represent the combination of a light and one push button, respectively.
References	:	BD/Rooms

Timed Function $curLightScene() \rightarrow LightScene$

Intention	:	This function represents the current light scene that is established by the machine if light scenes are enabled.
Scope	:	machine controlled, not visible for the environment
References	:	PD/U2, PD/U3, PD/U4
Comment	:	This entity is explicitly introduced by the customer.

Timed Function $curStandLightScene() \rightarrow LightScene$

Intention	:	This function represents the current standard (=default) light scene that is established by the machine, if light scenes are enabled.
Scope	:	machine controlled, not visible for the environment
References	:	PD/NF3

Timed Predicate *manualControl*

Intention	:	This predicate is true iff the light is controlled manually and not by the control system.
Scope	:	machine controlled, not visible for the environment
References	:	CustomerReviewResults

Timed Function $faultStandLightScene() \rightarrow LightScene$

Intention	: This function represents the light scene that has to become standard light scene in the case of a malfunction of the outdoor light sensors.
Scope	: machine controlled, not visible for the environment
References	: PD/NF3

Timed Function $noFaultStandLightScene() \rightarrow LightScene$

Intention	: This function represents the light scene that has to become standard light scene in the case that the outdoor light sensors have no malfunctions.
Scope	: machine controlled, not visible for the environment
References	: PD/U4

Timed Function $curAmbLightLev() \rightarrow LIGHT\_VALUE$

Intention	: This function represents the current ambient light level that is established by the machine if ambient light settings are enabled by the control panel.
Scope	: machine controlled, not visible for the environment
References	: PD/U9, PD/U14, PD/U16

Timed Function $defaultAmbLightLev() \rightarrow LIGHT\_VALUE$

Intention	: This function represents the default ambient light level.
References	: PD/U9

Function $evalLightScene(LightScene, LightScene :: LAMPS) \rightarrow PERCENT$

Intention	: This function represents the dimmer settings for a given lighting (represented by the second argument) that are necessary to achieve the ambient light intensity of the given light scene (represented by the first argument) under consideration of the light scene's lighting order.
Scope	: machine controlled, not visible for the environment
References	: CustomerReviewResults
Comment	: Note that 1, ..., 9 are forbidden dimmer settings. The specification of this function for an object of this class has to be provided by the domain expert. The domain expert has especially to say which are the entities this function depends on.

Function $evalAmbient(LIGHT\_VALUE, LightScene :: LAMPS) \rightarrow PERCENT$

Intention	: This function represents the dimmer setting for a given lighting (represented by the second argument) that is necessary to achieve the given ambient light intensity (represented by the first argument) together with the other lightings.
Scope	: machine controlled, not visible for the environment
References	: CustomerReviewResults
Comment	: Note that 1, ..., 9 are forbidden dimmer settings. The specification of this function for an object of this class has to be provided by the domain expert. The domain expert has especially to say which are the entities this function depends on.

Time Constant T_RL_1

Intention	: This time constant represents the maximal delay between occupation of a room, when a light scene is chosen, and the establishment of the chosen light scene.
References	: PD/U2, PD/U3, PD/U4

Time Constant T_RL_2

Intention	: This time constant represents the time during which the reoccupation of a room leads to reestablishing the previous light scene and the time delay between a sufficiently often occupation and the establishing of the chosen light scene.
References	: PD/U3, PD/U4, PD/U10

Requirements/Specifications

Initialization

Property I_RL_1

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

Property I_RL_2

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

Property I_RL_3

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

Property I_RL_4

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

END OF INITIALIZATION

Property P_RL_1

Formal	:	$\Box(noSaveLight \leftrightarrow manualControl \lor$ $cp.lCPRoom.lightSceneSwitch.checked = 1 \lor$ $cp.lCPRoom.ambientSwitch.checked = 1)$
NL	:	The light intensity in a room that is occupied can be less than the light intensity needed to establish *save light* iff the light is controlled manually or light scenes are enabled or the ambient light level is enabled.
References	:	PD/U1, CustomerReviewResults

Property P_RL_2

NonFormal	:	The standard light scene in the case that the outdoor light sensors have no malfunction is that the light intensity is sufficient to move safely.
References	:	PD/U1
Comment	:	The domain expert has to provide such a light scene that guarantees that the light intensity is sufficient to move safely. This light scene specifies the function *noFaultStandLightScene*

Property P_RL_3

NonFormal	:	The standard light scene in the case of a malfunction of the outdoor light sensors is that all ceiling lights are on.
References	:	PD/NF3
Comment	:	The domain expert has to provide such a light scene that guarantees that all ceiling lights are on. This light scene specifies the function *faultStandLightScene*.

Property P_RL_4

NonFormal	:	For each room a default light scene can be chosen (not by using the control panel).
References	:	PD/U8

Property P_RL_5

NonFormal	:	For each room a default ambient light level can be set (not by using the control panel).
References	:	PD/U9

Property P_RL_6

NonFormal	:	The value T_RL_2 can be set for each room separately (not by using the control panel).
References	:	PD/U10

Property P_RL_7

Formal	: $\Box(([occ.occUsed] \wedge \blacklozenge_{\leq T\_RL_2} occ.occUsed) \rightarrow curLightScene = cp.lCPRoom.chosenLightScene.checked)$
NL	: Whenever the room becomes occupied now and was occupied sometimes within the last T_RL_2 milliseconds and the light scene l was last chosen, the light scene l is the current light scene.
References	: PD/U3

Property P_RL_8

Formal	: $\Box([occ.occUsed] \wedge \blacksquare_{> T\_RL_2} \neg occ.occUsed) \rightarrow curLightScene = curStandLightScene)$
NL	: Whenever the room becomes occupied now and has not been occupied during the last time T_RL_2, the current light scene is the current standard light scene.
References	: PD/U4

Property P_RL_9

Formal	: $\Box(\forall l \in LightScenes : (occ.occUsed \wedge \neg[occ.occUsed] \wedge [cp.lCPRoom.chosenLightScene.checked = l]) \rightarrow \Diamond_{\leq T\_RL_1} curLightScene = l)$
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.
References	: PD/U2, PD/U1, PD/U6, PD/U12
Comment	: Perhaps it is domain knowledge that the two events *room becomes occupied ([occ.occUsed])* and *a current light scene becomes chosen ([cp.lCPRoom.chosenLightScene.checked = l])* can not happen together at the same time point. If this is domain knowledge the condition $\neg[occupied]$ can be deleted.

Property P_RL_9

Formal : $\Box((occ.occUsed \wedge \neg manualControl \wedge cp.lCPRoom.lightSceneSwitch.checked = 1) \rightarrow ((win.light.noMalSens.adjustedEntity = 1 \leftrightarrow evalLightScene(curLightScene, WINDOW) \neq 0) \wedge (win.dimmer.noMalSens.adjustedEntity = evalLightScene(curLightScene, WINDOW)) \wedge (wall.light.noMalSens.adjustedEntity = 1 \leftrightarrow evalLightScene(curLightScene, WALL) \neq 0) \wedge (wall.dimmer.noMalSens.adjustedEntity = evalLightScene(curLightScene, WALL))))$

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

Property P_RL_{10}

Formal : $\Box((occ.occUsed \wedge \neg manualControl \wedge cp.lCPRoom.lightSceneSwitch.checked = 0 \wedge cp.lCPRoom.ambientSwitch.checked) \rightarrow ((win.light.noMalSens.adjustedEntity = 1 \leftrightarrow evalAmbient(curAmbLightLev, WINDOW) \neq 0) \wedge (win.dimmer.noMalSens.adjustedEntity = evalAmbient(curAmbLightLev, WINDOW)) \wedge (wall.light.noMalSens.adjustedEntity = 1 \leftrightarrow evalAmbient(curAmbLightLev, WALL) \neq 0) \wedge (wall.dimmer.noMalSens.adjustedEntity = evalAmbient(curAmbLightLev, WALL))))$

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

Property P_RL_{11}

Formal : $\square(occ.occUsed \wedge \neg manualControl$
$\wedge\ cp.lCPRoom.lightSceneSwitch.checked = 0$
$\wedge\ \neg cp.lCPRoom.ambientSwitch.checked$
$\rightarrow ((win.light.noMalSens.adjustedEntity =$
$cp.lCPRoom.winCeilLightSwitch.checked) \wedge$
$(cp.lCPRoom.winCeilLightSwitch.checked = 1$
$\rightarrow win.dimmer.noMalSens.adjustedEntity = 100)$
$(cp.lCPRoom.winCeilLightSwitch.checked = 0$
$\rightarrow win.dimmer.noMalSens.adjustedEntity = 0)$
$(wall.light.noMalSens.adjustedEntity =$
$cp.lCPRoom.wallCeilLightSwitch.checked) \wedge$
$(cp.lCPRoom.wallCeilLightSwitch.checked = 1$
$\rightarrow wall.dimmer.noMalSens.adjustedEntity = 100)$
$(cp.lCPRoom.wallCeilLightSwitch.checked = 0$
$\rightarrow wall.dimmer.noMalSens.adjustedEntity = 0)))$

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

Property P_RL_{12}

Formal : $\square(manualControl$
$\rightarrow (\neg[win.light.noMalSens.adjustedEntity] \wedge$
$\neg[win.dimmer.noMalSens.adjustedEntity] \wedge$
$\neg[wall.light.noMalSens.adjustedEntity] \wedge$
$\neg[wall.dimmer.noMalSens.adjustedEntity]))$

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

Property P_RL_{13}

Formal : $\square(([occ.occUsed] \wedge \blacklozenge_{\leq T\_RL_2} occ.occUsed)$
$\rightarrow curAmbLightLev = cp.lCPRoom.ambLightLev.checked)$

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

References : PD/U3, CustomerReviewResults

Property P_RL_{14}

Formal	: $\Box(([occ.occUsed] \wedge \blacksquare_{>T\_RL_2} \neg occ.occUsed) \rightarrow curAmbLightLev = defaultAmbLightLev)$
NL	: Whenever the room becomes occupied now and has not been occupied during the last time T_RL_2, the current ambient light level is the default ambient light level.
References	: PD/U4, CustomerReviewResults

Property P_RL_{15}

Formal	: $\Box(\forall l \in LIGHT\_VALUES : (occ.occUsed \wedge \neg[occ.occUsed] \wedge [cp.lCPRoom.ambLightLev.checked = l]) \rightarrow \Diamond_{\leq T\_RL_1} curAmbLightLev = l)$
NL	: Whenever the room is occupied, but has not just become occupied now, and the ambient light level l becomes chosen, this ambient light level l is the current ambient light level within time T_RL_1.
References	: PD/U2, PD/U6, PD/U12, CustomerReviewResults
Comment	: Perhaps it is domain knowledge that the two events *room becomes occupied ([occ.occUsed])* and *a current light scene becomes chosen ([cp.lCPRoom.chosenLightScene.checked = l])* can not happen together at the same time point. If this is domain knowledge the condition $\neg[occupied]$ can be deleted.

Description Class RoomTemperature

Intention: This class provides all the definitions to describe the domain knowledge, the requirements, and the specifications concerning the temperature in a room.

Signature

Object *modes* : *TempOperationModeSet*

Intention	:	This object represents the operation modes of a room concerning temperature.

Object *adaptTimes* : *AdaptiveOccupancyTimes*

Intention	:	This object represents the delay times used to determine occupancy if the facility manager has chosen the delay times to be adaptive.

Object *cp* : *TemperatureCtrlPanel*

Intention	:	This object represents the temperature specific inputs of a user and facility manager.
Scope	:	environment controlled, not visible for the machine
References	:	BD/Rooms

Object *airTemp* : *RoomTempSensor*[3]

Intention	:	Each object represents a room temperature sensor of a room.
Scope	:	

Predicate *defaultTempOccDaytime*

Intention	:	This predicate represents the default value for occupancy concerning the temperature control during daytime in case of a malfunction of some sensors used to determine occupancy.
Scope	:	machine controlled, not visible for the environment
References	:	PD/NFH4
Comment	:	This entity is explicitly introduced by the customer. Note that this value can not be changed.

Predicate *defaultTempOccNighttime*

Intention	:	This predicate represents the default value for occupancy concerning the temperature control during nighttime in case of a malfunction of some sensors used to determine occupancy.
Scope	:	machine controlled, not visible for the environment
References	:	PD/NFH4
Comment	:	This entity is explicitly introduced by the customer. Note that this value can not be changed.

Requirements/Specifications

Property P_RT_1

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

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

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

References : PD/FMH5

Property P_RT_2

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

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

References : PD/UH1

Property P_RT_3

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

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

References : CustomerReviewResults

Property P_RT_4

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

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

References : CustomerReviewResults

Property P_RT_5

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

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

References : PD/FMH5

Property P_RT_6

Formal :
$$\Box((cp.tCPFM.adaptive.checked = 1 \rightarrow modes.occ.dt_1 = adaptTimes.dt_1Adaptive) \wedge (cp.tCPFM.adaptive.checked = 0 \rightarrow (modes.expUse.expectedUsage \rightarrow modes.occ.dt_1 = cp.expectedUsageTimes.dt_1.value) \wedge (\neg expectedUsage \rightarrow modes.occ.dt_1 = cp.notExpectedUsageTimes.dt_1.value)))$$

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

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

References : PD/FMH6, PD/Paragraph 7

Property P_RT_7

Formal : $\Box((cp.tCPFM.adaptive.checked = 1$
$\rightarrow modes.occ.dt_2 = adaptTimes.dt_2Adaptive) \wedge$
$(cp.tCPFM.adaptive.checked = 0 \rightarrow$
$(modes.expUse.expectedUsage \rightarrow$
$modes.occ.dt_2 = cp.expectedUsageTimes.dt_2.value) \wedge$
$(\neg modes.expUse.expectedUsage \rightarrow$
$modes.occ.dt_2 = cp.notExpectedUsageTimes.dt_2.value)))$

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

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

References : PD/FMH6, PD/Paragraph 7

Property P_RT_8

Formal	:	$\Box((cp.tCPFM.adaptive.checked = 1$ $\rightarrow modes.occ.dt_3 = adaptTimes.dt_3Adaptive) \wedge$ $(cp.tCPFM.adaptive.checked = 0 \rightarrow$ $(modes.expUse.expectedUsage \rightarrow$ $modes.occ.dt_3 = cp.expectedUsageTimes.dt_3.value) \wedge$ $(\neg modes.expUse.expectedUsage \rightarrow$ $modes.occ.dt_3 = cp.notExpectedUsageTimes.dt_3.value)))$
NL	:	The value *dt3* used for determining occupancy is specified as follows: • If the facility manager has chosen the adaptive mode for this room, *dt3* is the value computed by the adaptive learning function. • If the facility manager has *not* chosen the adaptive mode for this room and usage of this room is expected, *dt3* is the value as it is computed by the entered values of the user and the facility manager for the case that usage of the room is expected. • If the facility manager has *not* chosen the adaptive mode for this room and usage of this room is *not* expected, *dt3* is the value as it is computed by the entered values of the user and the facility manager for the case that usage of the room is *not* expected.
References	:	PD/FMH6, PD/Paragraph 7

Property P_RT_9

Formal	:	$\Box(defaultTempOccDaytime)$
NL	:	The default value for occupancy concerning the temperature control during daytime in case of a malfunction of some sensors used to determine occupancy is *occupancy*.
References	:	PD/NFH4

Property P_RT_{10}

Formal	:	$\Box(\neg defaultTempOccNighttime)$
NL	:	The default value for occupancy concerning the temperature control during nighttime in case of a malfunction of some sensors used to determine occupancy is *no occupancy*.
References	:	PD/NFH4

Property P_RT_{11}

Formal	:	$\Box(modes.occ.automaticDetermValue \leftrightarrow$ $(daytime \rightarrow defaultTempOccDaytime) \wedge$ $(\neg daytime \rightarrow defaultTempOccNighttime))$
NL	:	The value determined automatically by the machine in the case of a malfunction of some sensors used to determine occupancy concerning the temperature control depends on whether it is daytime or nighttime. The corresponding default values for the two time periods are used as the value determined automatically.
References	:	PD/NFH4

Property P_RT_{12}

NonFormal	:	The desired temperature shall be reached as fast as possible during the heating up state.
References	:	PD/UH2
Comment	:	The domain expert has to provide a function that realizes this requirement. This function has to provide at least for each time point a value for each radiator valve that should be adjusted.

Property P_RT_{13}

NonFormal	:	The currently desired temperature (represented by the function *roomDesiredTemperature*) is maintained during the current temperature state as good as possible.
References	:	PD/UH2
Comment	:	The domain expert has to provide a function that realizes this requirement. This function has to provide at least for each time point a value for each radiator valve that should be adjusted.

Description Class Room

Intention: This class provides all definitions to describe the behavior common to any kind of room.

Formal Parameters

Timed Function $numWinSect() \rightarrow NAT$

Intention	: This function represents the number of window sections that belong to a room.
References	: BD/Rooms

Signature

Object $RL : RoomLight$

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

Object $RT : RoomTemperature$

Intention	: This object represents the part that describes the beahvior of a room concerning the temperature control.

Object $ws : WindowSection[numWinSect]$

Intention	: Each object represents one of the window sections in a room.
Scope	: environment controlled, not visible for the machine
References	: BD/Rooms

Object $md : MotDet$

Intention	: This object represents the room motion detector.
Scope	: environment controlled, not visible for the machine
References	: BD/Rooms

Object $rt : RoomTempSensor[3]$

Intention	: Each object represents one of the three air temperature sensors in a room.
Scope	: environment controlled, not visible for the machine
References	: BD/Rooms

Timed Predicate $malfunctionAllTempSens$

Intention	: This predicate is true iff all temperature sensors of a room have a malfunction.
Scope	: machine controlled, not visible for the environment
References	: PD/NFH3
Comment	: This entity is explicitly introduced by the customer.

Timed Function $\mathit{malValvePos}() \rightarrow \mathit{THREE\_PT\_ACT\_RANGE}$

Intention	: This function represents the value determined for the radiator valves if all temperature sensors of a room have a malfunction.
Scope	: machine controlled, not visible for the environment
References	: PD/NFH3, PD/NFH9
Comment	: This entity is explicitly introduced by the customer. The specification of this function for an object of this class has to be provided by the domain expert.

Timed Function $\mathit{valueEstimation}(\mathit{ROOM\_TEMP}, \mathit{ROOM\_TEMP}) \rightarrow \mathit{ROOM\_TEMP}$

Intention	: This function represents the value determined for a room temperature sensor depending on the values of the other two room temperature sensors.
Scope	: machine controlled, not visible for the environment
Comment	: This entity is explicitly introduced by the customer. The specification of this function for an object of this class has to be provided by the domain expert. This function is used in the case of a malfunction of a room temperature sensor.

Timed Predicate $\mathit{malTemp}$

Intention	: This predicate is true iff the machine assumes that the sensors, used to determine the settings of the actuators controlling the temperature, have such a serious malfunction that the settings of the actuators have to be determined using makeshift operations.
Scope	: machine controlled, not visible for the environment
References	: PD/NFH3, NFH9

Timed Function $\mathit{tempRadSet}(\mathit{TEMP}) \rightarrow \mathit{THREE\_PT\_ACT\_RANGE}$

Intention	: This function represents the current setting of the radiators in a room intended by the temperature control.
Scope	: machine controlled, not visible for the environment

Timed Function $\mathit{tempRadNoMalSet}(\mathit{TEMP}) \rightarrow \mathit{THREE\_PT\_ACT\_RANGE}$

Intention	: This function represents the current setting of the radiators in a room intended by the temperature control if the machine assumes that there is no malfunction of the sensors needed to determine this setting. This setting depends on the current desired room temperature (first argument).
Scope	: machine controlled, not visible for the environment
References	: PD/UH2
Comment	: The specification of this function for an object of this class has to be provided by the domain expert.

Timed Function *tempRadMalSet*(*TEMP*, *TEMP*, *TEMP*, *TEMP*)
$\rightarrow$ *THREE_PT_ACT_RANGE*

Intention	: This function represents the current setting of the radiators in a room intended by the temperature control if the machine assumes that there is a malfunction of the sensors needed to determine this setting. This setting depends on the current desired room temperature (first argument), the current outdoor temperature (second argument), the settings of the neighbour rooms (third and fourth argument).
Scope	: machine controlled, not visible for the environment
References	: PD/NFH3, PD/NFH9
Comment	: This entity is explicitly introduced by the customer. The specification of this function for an object of this class has to be provided by the domain expert.

Requirements/Specifications

Initialization

Property I_R_1

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

End of Initialization

Property P_R_1

Formal	: $\square(rt[1].malSens.automaticDetermValue = valueEstimation(rt[2].noMalSens.convertedEntity, rt[3].noMalSens.convertedEntity))$
NL	: If the first room temperature sensor has a malfunction its value is estimated using the other two room temperature sensors.
References	: PD/NFH3

Property P_R_2

Formal	: $\square(rt[2].malSens.automaticDetermValue = valueEstimation(rt[1].noMalSens.convertedEntity, rt[3].noMalSens.convertedEntity))$
NL	: If the second room temperature sensor has a malfunction its value is estimated using the other two room temperature sensors.
References	: PD/NFH3

Property P_R_3

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

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

References : PD/NFH3

Property P_R_4

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

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

Property P_R_5

Formal : $\Box(malTemp \leftrightarrow malfunctionAllTempSens)$

NL : The machine assumes that the sensors, used to determine the settings of the actuators controlling the temperature, have such a serious malfunction that the settings of the actuators have to be determined using makeshift operations iff all temperature sensors of a room have a malfunction.

References : PD/NFH3

Property P_R_6

Formal : $\Box((\neg malTemp \rightarrow tempRadSet = tempRadNoMalSet) \wedge$
$(malTemp \rightarrow tempRadSet = tempRadMalSet))$

NL : If the machine does not assume a serious malfunction of the sensors, used to determine the settings of the actuators controlling the temperature, the current setting of the radiators in a room intended by the temperature control are as determined by the function *tempRadNoMalSet*.
If the machine assumes a serious malfunction of the sensors, used to determine the settings of the actuators controlling the temperature, the current setting of the radiators in a room intended by the temperature control are as determined by the function *tempRadMalSet*.

Property P_R_7

Formal	: $\Box(\forall n, 1 \leq n \leq numWinSect : ws[n].rad.valveAct.noMalAct.adjustedEntity = tempRadSet)$
NL	: The value adjusted by the machine for the radiator valves corresponds with the value intended by the temperature control for these actuators.
References	: PD/NFH3, PD/NFH9, CustomerReviewResults
Comment	: Note that all radiator valves in a room are always set to the same value. A specific value for a single radiator is not determined.

Description Class BlindedRoom

Intention: This class provides all definitions specific to any kind of room with sun blinds.

Base Classes

Class *Room*

Intention	: All definitions of Room are also provided here.

Signature

Object $Room :: ws : BlindedWindowSection[numWinSect]$

Intention	: The window sections in blinded rooms have sun blinds.
Scope	: environment controlled, not visible for the machine
References	: BD/Rooms, CustomerReviewResults

Timed Function $tempBlindSet(TEMP) \rightarrow THREE\_PT\_ACT\_RANGE$

Intention	: This function represents the current setting of the sun blinds in a room intended by the temperature control.
Scope	: machine controlled, not visible for the environment

Timed Function $tempBlindNoMalSet(TEMP) \rightarrow THREE\_PT\_ACT\_RANGE$

Intention	: This function represents the current setting of the sun blinds in a room intended by the temperature control if the machine assumes that there is no malfunction of the sensors needed to determine this setting. This setting depends on the current desired room temperature (first argument).
Scope	: machine controlled, not visible for the environment
Comment	: The specification of this function for an object of this class has to be provided by the domain expert.

Timed Function $tempBlindMalSet(TEMP) \rightarrow THREE\_PT\_ACT\_RANGE$

Intention	: This function represents the current setting of the sun blinds in a room intended by the temperature control if the machine assumes that there is a malfunction of the sensors needed to determine this setting. This setting depends on the current desired room temperature (first argument).
Scope	: machine controlled, not visible for the environment
Comment	: The specification of this function for an object of this class has to be provided by the domain expert.

Timed Function $lightBlindSet(TEMP) \rightarrow THREE\_PT\_ACT\_RANGE$

Intention	: This function represents the current setting of the sun blinds in a room intended by the light control.
Scope	: machine controlled, not visible for the environment

Timed Function $lightBlindNoMalSet(TEMP) \rightarrow THREE\_PT\_ACT\_RANGE$

Intention	: This function represents the current setting of the sun blinds in a room intended by the light control if the machine assumes that there is no malfunction of the sensors needed to determine this setting. This setting depends on the current desired room temperature (first argument).
Scope	: machine controlled, not visible for the environment
Comment	: The specification of this function for an object of this class has to be provided by the domain expert.

Timed Function $lightBlindMalSet(TEMP) \rightarrow THREE\_PT\_ACT\_RANGE$

Intention	: This function represents the current setting of the sun blinds in a room intended by the light control if the machine assumes that there is a malfunction of the sensors needed to determine this setting. This setting depends on the current desired room temperature (first argument).
Scope	: machine controlled, not visible for the environment
Comment	: The specification of this function for an object of this class has to be provided by the domain expert.

Timed Predicate $manualBlind$

Intention	: This predicate is true iff the sun blinds are controlled manually and not by the control system.
Scope	: machine controlled, not visible for the environment
References	: PD/UC2, CustomerReviewResults
Comment	: Note that either no sun blind or all sun blinds in a room are controlled manually. Manual control can not be enabled specifically for a single sun blind.

Domain Knowledge

Property D_BR_1

Formal	: $\Box(numWinSect = RL.cp.lCPRoom.numSunBlinds)$
NL	: Each window section is equipped with a sun blind.
References	: BD/Rooms, CustomerReviewResults

Requirements/Specifications

Initialization

Property I_BR_1

Formal	: $\neg manualBlind$
NL	: Initially the sun blinds are not controlled manually.

End of Initialization

Property P_BR_1

Formal	: $\Box(\neg RT.modes.occ.occUsed \rightarrow \neg manualBlind)$
NL	: If a room is not occupied according to the definition of this term used for temperature control the sun blinds are not controlled manually.
References	: CustomerReviewResults

Property P_R_2

Formal	: $\Box((\neg malTemp \rightarrow tempBlindSet = tempBlindNoMalSet) \wedge$ $(malTemp \rightarrow tempBlindSet = tempBlindMalSet))$
NL	: If the machine does not assume a serious malfunction of the sensors, used to determine the settings of the actuators controlling the temperature, the current setting of the sun blinds in a room intended by the temperature control are as determined by the function *tempBlindNoMalSet*. If the machine assumes a serious malfunction of the sensors, used to determine the settings of the actuators controlling the temperature, the current setting of the sun blinds in a room intended by the temperature control are as determined by the function *tempBlindMalSet*.

Property P_BR_3

Formal	: $\Box(manualBlind \rightarrow \forall s, 1 \leq s \leq numWinSect :$ $ws.sb[s].noMalAct.adjustedEntity =$ $RoomLight :: cp.sunBlindSwitch[s].checked)$
NL	: If the sun blinds are controlled manually the value adjusted by the machine for each sun blind is equal to the corresponding switch status in the control panel.
References	: PD/UH6, PD/UC1, CustomerReviewResults
Comment	: Note that manual control has a higher priority than machine control.

Property P_BR_4

Formal	: $\Box(\neg manualBlind \rightarrow \forall s, 1 \leq s \leq numWinSect :$ $ws.sb[s].noMalAct.adjustedEntity = tempBlindSet)$
NL	: If the sun blinds are not controlled manually the value adjusted by the machine for each sun blind is equal to the value determined for the sun blinds by the temperature control.
References	: PD/UC1, PD/UC2, CustomerReviewResults
Comment	: Note that manual control has a higher priority than machine control. The domain expert has to say in which way the intended settings of the light control concerning the sun blinds have to be considered. Up to now they are never established.

Description Class Office

Intention: This class provides all definitions specific to all offices having no sun blinds.

Base Classes

Class *Room*

Intention	:	All definitions of Room are also provided here.

Signature

Object tl : *LightButtonSet*

Intention	:	This Object represents the task light with its push button.
References	:	BD/Offices

Domain Knowledge

Property D_O_1

Formal	:	$\Box(allLightsOff \leftrightarrow win.light.envEntity = 0 \wedge wall.light.envEntity = 0 \wedge tl.envEntity = 0)$
NL	:	All lights are off if and only if both ceiling lights and the task light are off.
References	:	BD/Offices

Requirements/Specifications

Property P_O_1

NonFormal	:	The control panel should be installed moveably like a telephone in the offices.
References	:	PD/U13

Description Class NoOffice

Intention: This class provides all definitions specific to non-office rooms having no sun blinds.

Base Classes

Class *Room*

Intention	: All definitions of Room are also provided here.

Domain Knowledge

Property D_NO_1

Formal	: $\Box(allLightsOff \leftrightarrow win.light.envEntity = 0 \land wall.light.envEntity = 0)$
NL	: All lights are off if and only if both ceiling lights are off.
References	: BD/Rooms, BD/Computer Labs, BD/Meeting Rooms

Requirements/Specifications

Property P_NO_1

NonFormal	: The control panel should be installed near a door to the hallway.
References	: PD/U15

Description Class BlindedOffice

Intention: This class provides all definitions specific to all offices with sun blinds.

Base Classes

Class *BlindedRoom*

Intention	: All definitions of BlindedRoom are also provided here.

Signature

Object tl : *LightButtonSet*

Intention	: This object represents the task light with its push button.
References	: BD/Offices

Domain Knowledge

Property D_BO_1

Formal	: $\Box(RL.allLightsOff \leftrightarrow win.light.noMalSens.envEntity = 0 \wedge wall.light.noMalSens.envEntity = 0 \wedge tl.noMalSens.envEntity = 0)$
NL	: All lights are off if and only if both ceiling lights and the task light are off.
References	: BD/Offices

Requirements/Specifications

Property P_BO_1

NonFormal	: The control panel should be installed movable like a telephone in an office.
References	: PD/U13

Description Class BlindedNoOffice

Intention: This class provides all definitions specific to non-office rooms with sun blinds.

Base Classes

Class *BlindedRoom*

Intention	: All definitions of a BlindedRoom are also provided here.

Domain Knowledge

Property D_BNO_1

Formal	: $\Box(allLightsOff \leftrightarrow win.light.noMalSens.envEntity = 0 \wedge wall.light.noMalSens.envEntity = 0)$
NL	: All lights are off if and only if both ceiling lights are off.
References	: BD/Rooms, BD/Computer Labs, BD/Meeting Rooms

Requirements/Specifications

Property P_BNO_1

NonFormal	: The control panel should be installed near a door to the hallway.
References	: PD/U15

Description Class HallwaySection

Intention: This class provides all definitions to describe the hallway specific light requirements.
Note that there are no requirements concerning the temperature in a hallway.

Base Classes

Class *AreaLight*

Intention	: All definitions of AreaLight are also provided here.

Signature

Object *eastMd* : *MotDet*

Intention	: This object represents the motion detector in a hallway which is installed at the more eastern door of the hallway.
Scope	: environment controlled, not visible for the machine
References	: BD/Hallway

Object *westMd* : *MotDet*

Intention	: This object represents the motion detector in a hallway which is installed at the more western door of the hallway.
Scope	: environment controlled, not visible for the machine
References	: BD/Hallway

Object *middleMd* : *MotDet*

Intention	: This object represents the motion detector in a hallway which is installed in the middle of the hallway.
Scope	: environment controlled, not visible for the machine
References	: BD/Hallway

Object *light* : *LightButtonSet*

Intention	: This object represents the hallway light with it's corresponding set of push buttons.
Scope	: environment controlled, not visible for the machine
References	: BD/Hallway

Timed Predicate *nearHallwayDoor*

Intention	: This predicate is true iff a person is near a hallway door to this hallway section indicating that he wants to enter the hallway section.
Scope	: environment controlled, not visible for the machine
References	: PD/U18
Comment	: The specification of this predicate for an object of this class has to be provided by the domain expert.

Timed Predicate *hwLightUncontroll*

Intention	: This predicate is true iff the light is neither automatically nor manually controllable.
Scope	: environment controlled, not visible for the machine
References	: PD/NF6
Comment	: The definition of this predicate for an object of this class has to be provided by the domain expert.

Time Constant T_HS_1

Intention	: This constant represents the time delay between a time point a person is near a hallway door indicating that he wants to enter this hallway section and switching on the lights.
References	: PD/U18

Time Constant T_HS_2

Intention	: This constant represents the time delay between the loss of light control in this hallway section and switching on the lights.

Requirements/Specifications

Property P_HS_1

Formal	: $\Box(\neg noSaveLight)$
NL	: The light intensity in a hallway section that is occupied has always to be greater than or equal to the light intensity needed to establish *save light*.
References	: PD/U17

Property P_HS_2

Formal	: $\Box(nearHallwayDoor \Rightarrow_{\leq T\_HS_1} light.noMalSens.envEntity = 1)$
NL	: Whenever a person is continuously in the near of a hallway door of the hallway section for at least $\mathrm{T\_HS}_1$ time units, then the light in the hallway section will be on within this time span, and remains on at least as long as the person is near the hallway section.
References	: PD/U18

Property P_HS_3

Formal	: $\Box(hwLightUncontroll \Rightarrow_{\leq T\_HS_2} light.noMalSens.envEntity = 1)$
NL	: Whenever the lights are neither controllable automatically nor manually for at least $\mathrm{T\_HS}_2$ time units, then eventually within this time span, the light in the hallway section is on and remains so at least as long as the lights are neither controllable automatically nor manually.
References	: PD/NF6

Description Class StairCase

Intention: This class provides all definitions to describe the entities belonging to a staircase which are relevant for the considered system.

SIGNATURE

Object *md* : *MotDet*

Intention	: This object represents a motion detector placed in a staircase in front of the hallway door.
Scope	: environment controlled, not visible for the machine
References	: BD/Staircase

Main Description Class Floor

Intention: This class is the main class. It contains all the objects belonging to the considered system, the domain knowledge and the requirements/specifications of the system that should be developed.

Signature

Object $o435 : \mathit{Office}[\mathit{numWinSect} = 2]$

Intention	:	This object represents the office in floor 32/4 with No 435 having two window sections without sun blinds.
Scope	:	environment controlled, not visible for the machine
References	:	BD/Figure 1, BD/Sections

Object $o433, o431, o429, o425, o423, o419, o417, o415 : \mathit{Office}[\mathit{numWinSect} = 2]$

Intention	:	These objects represent the corresponding offices having two window sections without sun blinds.
Scope	:	environment controlled, not visible for the machine
References	:	BD/Figure 1, CustomerReviewResults

Object $o421 : \mathit{Office}[\mathit{numWinSect} = 3]$

Intention	:	This object represents the office in floor 32/4 with No 421 having three window sections without sun blinds.
Scope	:	environment controlled, not visible for the machine
References	:	BD/Figure 1, CustomerReviewResults

Object $o424, o416, o414, o412 : \mathit{BlindedOffice}[\mathit{numWinSect} = 2]$

Intention	:	These objects represent the corresponding offices with sun blinds.
Scope	:	environment controlled, not visible for the machine
References	:	BD/Figure 1, CustomerReviewResults

Object $m427 : \mathit{NoOffice}[\mathit{numWinSect} = 5]$

Intention	:	This object represents the meeting room M427 having five window sections without sun blinds.
References	:	BD/Figure 1, BD/Sections, CustomerReviewResults

Object $cl411 : \mathit{NoOffice}[\mathit{numWinSect} = 9]$

Intention	:	This object represents the computer lab CL411 having nine window sections without sun blinds.
References	:	BD/Figure 1, BD/Sections, CustomerReviewResults

Object $cl426, cl422, cl418 : \mathit{BlindedNoOffice}[\mathit{numWinSect} = 3]$

Intention	:	These objects represent the corresponding rooms which are no offices having three window sections with sun blinds.
References	:	BD/Figure 1, BD/Sections, CustomerReviewResults

Object $cl410 : BlindedNoOffice[numWinSect = 10]$

Intention : This object represents computer lab CL410 having ten window sections with sun blinds.
References : BD/Figure 1, BD/Sections, CustomerReviewResults

Domain $ROOMS = \{o435, o433, o432, o429, o425, o423, o421, o419, o417, o415, o424, o416, o414, o412, p413, m427, cl411, cl426, cl422, cl418, cl410\}$

Intention : This domain contains all the rooms that have to be considered.
References : BD/Figure 1, BD/Sections

Domain $OFFICES = \{o435, o433, o431, o429, o425, o423, o421, o419, o417, o415, o424, o416, o414, o412\}$

Intention : This domain contains all the offices that have to be considered.
References : BD/Figure 1, BD/Sections

Domain $MeetingRooms = \{m427\}$

Intention : This domain contains all the meeting rooms that have to be considered.
References : BD/Figure 1, BD/Sections

Domain $ComputerLabs = \{cl411, cl426, cl422, cl418, cl410\}$

Intention : This domain contains all the computer labs that have to be considered.
References : BD/Figure 1, BD/Sections

Domain $PeripheralRooms = \{p413\}$

Intention : This domain contains all the peripheral rooms that have to be considered.
References : BD/Figure 1, BD/Sections

Object $d\_421\_423 : Door$

Intention : This object represents the door between the offices o421 and o423.
References : BD/Figure 1

Object $d\_423\_425 : Door$

Intention : This object represents the door between the offices o423 and o425.
References : BD/Figure 1

Object $d\_431\_433 : Door$

Intention : This object represents the door between the offices o431 and o433.
References : BD/Figure 1

Object d_433_435 : *Door*

Intention	: This object represents the door between the offices o433 and o435.
References	: BD/Figure 1

Object d_414_416 : *Door*

Intention	: This object represents the door between the offices o414 and o416.
References	: BD/Figure 1

Object d_416_418 : *Door*

Intention	: This object represents the door between the offices o416 and o418.
References	: BD/Figure 1

Object d_418_420 : *Door*

Intention	: This object represents the door between the offices o418 and o420.
References	: BD/Figure 1

Object d_420_422 : *Door*

Intention	: This object represents the door between the offices o420 and o422.
References	: BD/Figure 1

Object d_o435, d_o433, d_o432, d_o429, d_o425, d_o423, d_o421, d_o419, d_o417, d_o415, d_o424, d_o416, d_o414, d_o412, d_m427, d_cl411, d_cl426, d_cl422, d_cl418, d_cl410 : *Door*

Intention	: These objects represent the corresponding doors of the rooms to the hallway.
References	: BD/Figure 1

Object $h1$: *HallwaySection*

Intention	: This object represents the hallway in the east of floor 32/4.
References	: BD/Sections, BD/Figure 1 (H1)

Object $h2$: *HallwaySection*

Intention	: This object represents the hallway in the middle of floor 32/4.
References	: BD/Sections, BD/Figure 1 (H2)

Object $h3$: *HallwaySection*

Intention	: This object represents the hallway in the west of floor 32/4.
References	: BD/Sections, BD/Figure 1 (H3)

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

Intention	: This domain contains all the hallway sections that have to be considered.
References	: BD/Figure 1

Object *ols1* : *OutdoorLightSensor*

Intention	: This object represents the outdoor light sensor in the north-east direction.
References	: BD/Sections, BD/Figure 1 (ols1)

Object *ols2* : *OutdoorLightSensor*

Intention	: This object represents the outdoor light sensor in the south-west direction.
References	: BD/Sections, BD/Figure 1 (ols2)

Object *ols3* : *OutdoorLightSensor*

Intention	: This object represents the outdoor light sensor in the north-west direction.
References	: BD/Sections, BD/Figure 1 (ols3)

Object *ols4* : *OutdoorLightSensor*

Intention	: This object represents the outdoor light sensor in the south-east direction.
References	: BD/Sections, BD/Figure 1 (ols4)

Object *ols5* : *OutdoorLightSensor*

Intention	: This object represents the outdoor light sensor in the north direction.
References	: BD/Sections, BD/Figure 1 (ols5)

Object *ols6* : *OutdoorLightSensor*

Intention	: This object represents the outdoor light sensor in the south direction.
References	: BD/Sections, BD/Figure 1 (ols6)

Object *ots* : *OutdoorTempSensor*

Intention	: This object represents the outdoor temperature sensor.
Scope	: environment controlled, not visible for the machine
References	: BD/Actuators: outdoor temperature sensor

Object *scw* : *StairCase*

Intention	: This object represents the staircase in the west of floor 32/4.
References	: BD/Sections, BD/Figure 1 (SCW)

Object *sce* : *StairCase*

Intention	: This object represents the staircase in the east of floor 32/4.
References	: BD/Sections, BD/Figure 1 (SCE)

Time Constant $T_{malReact}$

Intention	: The time delay between a malfunction of the outdoor light sensor and switching on the lights if the hallway is occupied.
Scope	:

Timed Predicate *outLightSensorFaultMach*

Intention	: This predicate is true iff the machine assumes that some outdoor light sensors have a malfunction.
Scope	: machine controlled, not visible for the environment
References	: PD/NF2, PD/NF3, PD/NF4
Comment	: Note that this predicate represents the situation as it is seen by the machine.

Requirements/Specifications

Property P_F_1

Formal : $\Box(\mathit{outLightSensorFaultMach} \leftrightarrow \mathit{ols1.malSens.malDetMach} \vee \mathit{ols2.malSens.malDetMach} \vee \mathit{ols3.malSens.malDetMach} \vee \mathit{ols4.malSens.malDetMach} \vee \mathit{ols5.malSens.malDetMach} \vee \mathit{ols6.malSens.malDetMach})$

NL : Some outdoor light sensor has a malfunction. This is a specification, i.e. it describes the situation as it should be seen by the machine.

Property P_F_2

Formal : $\Box(\forall r \in \mathit{ROOMS} : \mathit{outLightSensorFaultMach} \rightarrow r.\mathit{curStandLightScene} = r.\mathit{faultStandLightScene})$

NL : Whenever the machine assumes that some outdoor light sensors are not working correctly, then the standard light scene is the standard fault light scene, i.e. all ceiling lights are on.

References : PD/NF3

Comment : Note that this specification does not specify that the standard fault light scene has to be established, i.e. that all ceiling lights have to be on. This specification only specifies what the standard light scene has to be in the case of a malfunction of some outdoor light sensors.

Property P_F_3

Formal : $\Box(\forall h \in \mathit{HALLWAYS} : \mathit{outLightSensorFaultMach} \wedge h.\mathit{occ.occUsed} \Rightarrow_{\leq T_{malReact}} h.\mathit{light.light.noMalSens.envEntity} = 1)$

NL : Whenever some outdoor light sensors are not working correctly and the hallway is occupied for at least $T_{malReact}$ time units, then eventually within this time span, the light in the hallway is on and remains on at least as long as the precondition is true.

References : PD/NF4

Property P_F_4

NonFormal	:	Energy consumption should be reduced as far as possible during maintaining of comfortable working conditions for all users.
References	:	PD/FMH1
Comment	:	The domain expert has to consider this requirement when specifying the control algorithms.

Property P_F_5

NonFormal	:	The use of solar radiation for heating during all times if heating is required or not disallowed, and the use of storage capacity of the building should always be preferred against the usage of the central heating unit.
References	:	PD/FMH2
Comment	:	The domain expert has to consider this requirement when specifying the control algorithms.

Property P_F_6

NonFormal	:	If a malfunction occurs, the control system supports the facility manager by finding the reason.
References	:	PD/FM8, CustomerReviewResults
Comment	:	According to the customer it is sufficient to mention the sensor or actuator which has a malfunction. Such information has to be provided in the mail sent in the case of a malfunction of a sensor or an actuator.

Property P_F_7

NonFormal	:	The system provides reports on current and past energy consumption.
References	:	PD/FM9, PD/FMH11
Comment	:	The domain expert has to provide a function to compute the energy consumption.

Property P_F_8

NonFormal	:	All malfunctions and unusual conditions are stored and reported on request
References	:	PD/FM10, PD/FMH12
Comment	:	This has to be realized during the implementation. Note that only malfunctions have to be stored and reported on request. There are up to now no alarms or unusual conditions specified by the customer or the domain expert.

Property P_F_9

NonFormal	:	All hardware connections have to be made according to DIN standards.
References	:	PD/NF7, PD/NFH5

Property P_F_{10}

NonFormal	: No hazardous condition for persons, inventory or building are allowed.
References	: PD/NF8, PD/NFH6

Property P_F_{11}

NonFormal	: The system warns about unreasonable inputs.
References	: PD/NF10

Property P_F_{12}

NonFormal	: If the whole control system has a serious malfunction, complete manually operation should be possible.
References	: PD/NFH2
Comment	: If manual operation is desired appropriate devices have to be installed.

Property P_F_{13}

NonFormal	: The control system shall be designed such that it can be easily changed and expanded.
References	: PD/NFH8

Property P_F_{14}

NonFormal	: The control system provides basic energy saving procedures automatically for control comfort. If the facility manager deviates from such settings, the system provides a report on expected results.
References	: PD/FMH9
Comment	: The domain expert has to provide the *basic energy saving procedures.*

Index

H

I

L

M

A References

BD/Actuators: above the figure in
Description Class DimmableLightButtonSet, DomKnow, *Property* D_DLBS_2
Description Class DimmableLightButtonSet, DomKnow, *Property* D_DLBS_1
Description Class DimmableLightButtonSet, Signature, *Object LightButtonSet* :: *light*
Description Class DimmableLight, Base Classes, *Class Light*
Description Class DimmableLight, DomKnow, *Property* D_DL_1
Description Class DimmableLight, DomKnow, *Property* D_DL_2
Description Class DimmableLight, Signature, *Object conSysActive*
Description Class DimmableLight, Signature, *Object dimmer*
Description Class DimmableLight, Signature, *Object pulse*
Description Class DimmableLight, Signature, *Time Constant safeModeReactDelay*
Description Class DimmableLight, Signature, *Timed Predicate safeMode*
Description Class ToggleActuator, DomKnow, *Property* D_TA_1
Description Class ToggleActuator, DomKnow, *Property* D_TA_2

BD/Actuators: blind lifter/rotator in
Description Class SunBlind, DomKnow, *Property* D_SBL_{10}
Description Class SunBlind, DomKnow, *Property* D_SBL_3
Description Class SunBlind, DomKnow, *Property* D_SBL_4
Description Class SunBlind, DomKnow, *Property* D_SBL_6
Description Class SunBlind, DomKnow, *Property* D_SBL_7
Description Class SunBlind, DomKnow, *Property* D_SBL_8
Description Class SunBlind, DomKnow, *Property* D_SBL_9
Description Class SunBlind, Signature, *Domain DEGREE*
Description Class SunBlind, Signature, *Timed Function rotateStatus*
Description Class SunBlind, Signature, *Timed Predicate moveDown*
Description Class SunBlind, Signature, *Timed Predicate moveStop*
Description Class SunBlind, Signature, *Timed Predicate moveUp*
Description Class SunBlind, Signature, *Timed Predicate rotateDecrease*
Description Class SunBlind, Signature, *Timed Predicate rotateIncrease*
Description Class SunBlind, Signature, *Timed Predicate rotateStop*
Global Signature, *Domain PERCENT*

BD/Actuators: dimmable light (cia) in
Description Class ControlSystemActive, DomKnow, *Property* D_CSA_1
Global Signature, *Domain BINARY*

BD/Actuators: dimmable light (dll) in
Description Class Dimmer, DomKnow, *Property* D_DIM_1
Global Signature, *Domain PERCENT*

BD/Actuators: dimmable light (sll) in
Description Class Light, DomKnow, *Property* D_L_1
Global Signature, *Domain BINARY*

BD/Actuators: figure in
Description Class DimmableLight, Base Classes, *Class Light*
Description Class DimmableLight, Signature, *Object conSysActive*
Description Class DimmableLight, Signature, *Object dimmer*
Description Class DimmableLight, Signature, *Object pulse*
Description Class Light, Base Classes, *Class BinarySensor*

BD/Actuators: introduction in
Description Class NoMalfunctionActuator, Signature, *Time Constant reactionTime*
Description Class NoMalfunctionActuator, Signature, *Timed Function adjustedEntity*

Main Description Class Floor, Signature, *Object d_414_416*
Main Description Class Floor, Signature, *Object d_416_418*
Main Description Class Floor, Signature, *Object d_418_420*
Main Description Class Floor, Signature, *Object d_420_422*
Main Description Class Floor, Signature, *Object d_421_423*
Main Description Class Floor, Signature, *Object d_423_425*
Main Description Class Floor, Signature, *Object d_431_433*
Main Description Class Floor, Signature, *Object d_433_435*
Main Description Class Floor, Signature, *Object d_o435*,
Main Description Class Floor, Signature, *Object m427*
Main Description Class Floor, Signature, *Object o421*
Main Description Class Floor, Signature, *Object o424*,
Main Description Class Floor, Signature, *Object o433*,
Main Description Class Floor, Signature, *Object o435*

BD/Hallway in
Description Class HallwaySection, Signature, *Object eastMd*
Description Class HallwaySection, Signature, *Object light*
Description Class HallwaySection, Signature, *Object middleMd*
Description Class HallwaySection, Signature, *Object westMd*
Description Class LightButtonSet, DomKnow, *Property* D_LBS_5
Description Class LightButtonSet, DomKnow, *Property* D_LBS_4
Description Class LightButtonSet, DomKnow, *Property* D_LBS_2
Description Class LightButtonSet, DomKnow, *Property* D_LBS_3
Description Class LightButtonSet, Signature, *Object light*
Description Class LightButtonSet, Signature, *Timed Predicate pushButton*

BD/Meeting Rooms in
Description Class BlindedNoOffice, DomKnow, *Property* D_BNO_1
Description Class NoOffice, DomKnow, *Property* D_NO_1

BD/Offices in
Description Class BlindedOffice, DomKnow, *Property* D_BO_1
Description Class BlindedOffice, Signature, *Object tl*
Description Class DimmableLight, Base Classes, *Class Light*
Description Class DimmableLight, Signature, *Object dimmer*
Description Class LightButtonSet, DomKnow, *Property* D_LBS_5
Description Class LightButtonSet, DomKnow, *Property* D_LBS_4
Description Class LightButtonSet, DomKnow, *Property* D_LBS_2
Description Class LightButtonSet, DomKnow, *Property* D_LBS_3
Description Class LightButtonSet, Signature, *Object light*
Description Class LightButtonSet, Signature, *Timed Predicate pushButton*
Description Class Light, Base Classes, *Class BinarySensor*
Description Class Office, DomKnow, *Property* D_O_1
Description Class Office, Signature, *Object tl*

BD/Rooms in
Description Class BlindedNoOffice, DomKnow, *Property* D_BNO_1
Description Class BlindedRoom, DomKnow, *Property* D_BR_1
Description Class BlindedRoom, Signature, *Object Room :: ws*
Description Class BlindedWindowSection, Signature, *Object sb*
Description Class DimmableLightButtonSet, DomKnow, *Property* D_DLBS_2
Description Class DimmableLightButtonSet, DomKnow, *Property* D_DLBS_1
Description Class DimmableLightButtonSet, Signature, *Object LightButtonSet :: light*
Description Class LightButtonSet, DomKnow, *Property* D_LBS_5
Description Class LightButtonSet, DomKnow, *Property* D_LBS_4

Description Class BinaryActuator, DomKnow, *Property* D_BA_1
Description Class BinarySensor, DomKnow, *Property* D_BS_1
Description Class BinarySensor, DomKnow, *Property* D_BS_2
Description Class BinarySensor, DomKnow, *Property* D_BS_3
Description Class BlindedRoom, DomKnow, *Property* D_BR_1
Description Class BlindedRoom, ReqSpec, *Property* P_BR_1
Description Class BlindedRoom, ReqSpec, *Property* P_BR_3
Description Class BlindedRoom, ReqSpec, *Property* P_BR_4
Description Class BlindedRoom, Signature, *Object Room* :: *ws*
Description Class BlindedRoom, Signature, *Timed Predicate manualBlind*
Description Class CalendarInterval, ReqSpec, *Property* P_CI_1
Description Class ControlledInput, ReqSpec, *Property* I_CIN_1
Description Class ControlledInput, ReqSpec, *Property* I_CIN_2
Description Class ControlledInput, ReqSpec, *Property* P_CIN_1
Description Class ControlledInput, ReqSpec, *Property* P_CIN_3
Description Class DimmableLight, DomKnow, *Property* D_DL_3
Description Class DimmableLight, Signature, *Object pulse*
Description Class DynamicInterval, ReqSpec, *Property* P_DI_1
Description Class DynamicInterval, Signature, *Predicate leq*
Description Class EnteredValue, Signature, *Timed Function entered*
Description Class ExpectedUsage, ReqSpec, *Property* D_EU_1
Description Class LightButtonSet, DomKnow, *Property* D_LBS_1
Description Class LightCtrlPanelFM, Signature, *Object lightOn*
Description Class LightCtrlPanelRoom, Signature, *Object ambientSwitch*
Description Class LightCtrlPanelRoom, Signature, *Object lightSceneSwitch*
Description Class LightCtrlPanelRoom, Signature, *Object sunBlindSwitch*[
Description Class LightCtrlPanelRoom, Signature, *Object wallCeilLightSwitch*
Description Class LightCtrlPanelRoom, Signature, *Object winCeilLightSwitch*
Description Class LightSceneOffice, ReqSpec, *Property* P_LSO_1
Description Class LightScene, ReqSpec, *Property* P_LS_1
Description Class MalfunctionActuator, ReqSpec, *Property* I_MA_1
Description Class MalfunctionActuator, ReqSpec, *Property* I_MA_2
Description Class MalfunctionActuator, ReqSpec, *Property* P_MA_1
Description Class MalfunctionActuator, ReqSpec, *Property* P_MA_2
Description Class MalfunctionActuator, ReqSpec, *Property* P_MA_3
Description Class MalfunctionActuator, ReqSpec, *Property* P_MA_4
Description Class MalfunctionActuator, ReqSpec, *Property* P_MA_5
Description Class MalfunctionActuator, ReqSpec, *Property* P_MA_6
Description Class MalfunctionActuator, Signature, *Timed Predicate FMInformed*
Description Class MalfunctionActuator, Signature, *Timed Predicate malDetEnv*
Description Class MalfunctionActuator, Signature, *Timed Predicate malDetMach*
Description Class MalfunctionActuator, Signature, *Timed Predicate userInformed*
Description Class MalfunctionOccupancy, ReqSpec, *Property* I_MO_1
Description Class MalfunctionOccupancy, ReqSpec, *Property* P_MO_1
Description Class MalfunctionSensor, ReqSpec, *Property* I_MS_2
Description Class MalfunctionSensor, ReqSpec, *Property* I_MS_3
Description Class MalfunctionSensor, ReqSpec, *Property* I_MS_1
Description Class MalfunctionSensor, ReqSpec, *Property* P_MS_2
Description Class MalfunctionSensor, ReqSpec, *Property* P_MS_3
Description Class MalfunctionSensor, ReqSpec, *Property* P_MS_4
Description Class MalfunctionSensor, ReqSpec, *Property* P_MS_5
Description Class MalfunctionSensor, ReqSpec, *Property* P_MS_6
Description Class MalfunctionSensor, ReqSpec, *Property* P_MS_7
Description Class MalfunctionSensor, ReqSpec, *Property* P_MS_1

Description Class MalfunctionSensor, Signature, *Timed Predicate FMInformed*
Description Class MalfunctionSensor, Signature, *Timed Predicate malDetEnv*
Description Class MalfunctionSensor, Signature, *Timed Predicate malDetMach*

PD/FM8 in
Description Class MalfunctionSensor, Signature, *Timed Predicate malDetEnv*
Description Class MalfunctionSensor, Signature, *Timed Predicate malDetMach*
Main Description Class Floor, ReqSpec, *Property* P_F_6

PD/FM9 in
Main Description Class Floor, ReqSpec, *Property* P_F_7

PD/FMH1 in
Main Description Class Floor, ReqSpec, *Property* P_F_4

PD/FMH10 in
Description Class ControlledInput, ReqSpec, *Property* P_CIN_3
Description Class ControlledInput, Signature, *Object allowedRange*

PD/FMH11 in
Main Description Class Floor, ReqSpec, *Property* P_F_7

PD/FMH12 in
Description Class MalfunctionSensor, Signature, *Timed Predicate malDetEnv*
Description Class MalfunctionSensor, Signature, *Timed Predicate malDetMach*
Main Description Class Floor, ReqSpec, *Property* P_F_8

PD/FMH13 in
Description Class Actuator, Signature, *Object manualEnteredMal*
Description Class Actuator, Signature, *Timed Predicate stateManualEnteredMal*
Description Class Occupancy, Signature, *Object manualEnteredMal*
Description Class Occupancy, Signature, *Timed Predicate stateManualEnteredMal*
Description Class Sensor, Signature, *Object manualEnteredMal*
Description Class Sensor, Signature, *Timed Predicate stateManualEnteredMal*

PD/FMH2 in
Main Description Class Floor, ReqSpec, *Property* P_F_5

PD/FMH3 in
Description Class ControlledInput, ReqSpec, *Property* P_CIN_2
Description Class ControlledInput, Signature, *Object allowedRange*
Description Class ControlledInput, Signature, *Object default*
Description Class ControlledInput, Signature, *Timed Function value*
Description Class ControlledInput, Signature, *Timed Predicate userValueEntered*
Description Class DynamicInterval, Signature, *Object end*
Description Class DynamicInterval, Signature, *Object start*
Description Class EnteredValue, ReqSpec, *Property* P_EV_1
Description Class EnteredValue, Signature, *Timed Function checked*
Description Class EnteredValue, Signature, *Timed Function entered*
Description Class EnteredValue, Signature, *Timed Function nearestValue*
Description Class ExpectedUsage, Signature, *Object holidayTime*
Description Class ExpectedUsage, Signature, *Object workingTime*
Description Class TemperatureCtrlPanel, Signature, *Object comfortTemperature*

PD/FMH4 in
Description Class CalendarInterval, Base Classes, *Class DynamicInterval*
Description Class CalendarInterval, Signature, *Timed Predicate actual*
Description Class DynamicInterval, Signature, *Object end*
Description Class DynamicInterval, Signature, *Object start*
Description Class EnteredValue, Signature, *Timed Function entered*

Description Class TemperatureCtrlPanelFM, Signature, *Object heatingPeriod*
Description Class TemperatureCtrlPanelFM, Signature, *Object nonHeatingPeriod*
Global Signature, *Domain CALENDAR_TIME*

PD/FMH5 in
Description Class EnteredValue, Signature, *Timed Function entered*
Description Class RoomTemperature, ReqSpec, *Property* P_RT_1
Description Class RoomTemperature, ReqSpec, *Property* P_RT_5
Description Class TempOperationModeSet, ReqSpec, *Property* P_TOMS_8
Description Class TempOperationModeSet, Signature, *Object NonFreezeMode*
Description Class TempOperationModeSet, Signature, *Timed Function roomDesiredTemperature*
Description Class TempOperationModeSet, Signature, *Timed Predicate nonFreezeTime*
Description Class TempOperationMode, Signature, *Timed Function desiredTemperature*
Description Class TemperatureCtrlPanelFM, Signature, *Object buildingNonFreezePeriod*
Description Class TemperatureCtrlPanelFM, Signature, *Object nonFreezePeriod*
Description Class TemperatureCtrlPanelFM, Signature, *Object nonFreezeTemperature*
Description Class TemperatureCtrlPanelFM, Signature, *Timed Predicate roomNonFreezeDefined*
Global Signature, *Domain CALENDAR_TIME*

PD/FMH6 in
Description Class AdaptiveOccupancyTimes, Signature, *Timed Function* $dt_1Adaptive$
Description Class AdaptiveOccupancyTimes, Signature, *Timed Function* $dt_2Adaptive$
Description Class AdaptiveOccupancyTimes, Signature, *Timed Function* $dt_3Adaptive$
Description Class EnteredValue, Signature, *Timed Function entered*
Description Class RoomTemperature, ReqSpec, *Property* P_RT_6
Description Class RoomTemperature, ReqSpec, *Property* P_RT_7
Description Class RoomTemperature, ReqSpec, *Property* P_RT_8
Description Class TemperatureCtrlPanelFM, Signature, *Object adaptive*

PD/FMH7 in
Description Class MalfunctionSensor, ReqSpec, *Property* P_MS_5
Description Class MalfunctionSensor, Signature, *Time Constant* T_{info}
Description Class MalfunctionSensor, Signature, *Time Constant* $T_{infDelay}$
Description Class MalfunctionSensor, Signature, *Timed Predicate FMInformed*

PD/FMH9 in
Main Description Class Floor, ReqSpec, *Property* P_F_{14}

PD/NF1 in
Description Class MalfunctionSensor, Signature, *Timed Function valueMal*
Description Class Sensor, Signature, *Timed Function usedValue*

PD/NF10 in
Description Class ControlledInput, Signature, *Object allowedRange*
Main Description Class Floor, ReqSpec, *Property* P_F_{11}

PD/NF2 in
Description Class MalfunctionSensor, Signature, *Timed Function automaticDetermValue*
Description Class MalfunctionSensor, Signature, *Timed Function valueMal*
Description Class MalfunctionSensor, Signature, *Timed Predicate malDetEnv*
Description Class MalfunctionSensor, Signature, *Timed Predicate malDetMach*
Description Class OutdoorLightSensor, ReqSpec, *Property* P_OLS_1
Description Class OutdoorLightSensor, Signature, *Timed Function lastCorrectOutLight*
Description Class Sensor, Signature, *Timed Function usedValue*
Main Description Class Floor, Signature, *Timed Predicate outLightSensorFaultMach*

PD/NF3 in
Description Class MalfunctionSensor, Signature, *Timed Function automaticDetermValue*
Description Class MalfunctionSensor, Signature, *Timed Function valueMal*

Description Class MalfunctionSensor, Signature, *Timed Predicate malDetEnv*
Description Class MalfunctionSensor, Signature, *Timed Predicate malDetMach*
Description Class RoomLight, ReqSpec, *Property* P_RL_3
Description Class RoomLight, Signature, *Timed Function curStandLightScene*
Description Class RoomLight, Signature, *Timed Function faultStandLightScene*
Description Class Sensor, Signature, *Timed Function usedValue*
Main Description Class Floor, ReqSpec, *Property* P_F_2
Main Description Class Floor, Signature, *Timed Predicate outLightSensorFaultMach*

PD/NF4 in
Description Class MalfunctionSensor, Signature, *Timed Function automaticDetermValue*
Description Class MalfunctionSensor, Signature, *Timed Function valueMal*
Description Class MalfunctionSensor, Signature, *Timed Predicate malDetEnv*
Description Class MalfunctionSensor, Signature, *Timed Predicate malDetMach*
Description Class Sensor, Signature, *Timed Function usedValue*
Main Description Class Floor, ReqSpec, *Property* P_F_3
Main Description Class Floor, Signature, *Timed Predicate outLightSensorFaultMach*

PD/NF5 in
Description Class AreaLight, ReqSpec, *Property* P_AL_2
Description Class MalfunctionOccupancy, ReqSpec, *Property* P_MO_1
Description Class MalfunctionOccupancy, Signature, *Timed Predicate automaticDetermValue*
Description Class MalfunctionOccupancy, Signature, *Timed Predicate malOccDetEnv*
Description Class MalfunctionOccupancy, Signature, *Timed Predicate malOccDetMach*
Description Class MalfunctionOccupancy, Signature, *Timed Predicate occMal*
Description Class MalfunctionSensor, Signature, *Timed Function automaticDetermValue*
Description Class MalfunctionSensor, Signature, *Timed Function valueMal*
Description Class MalfunctionSensor, Signature, *Timed Predicate malDetEnv*
Description Class MalfunctionSensor, Signature, *Timed Predicate malDetMach*
Description Class Occupancy, Signature, *Timed Predicate occUsed*
Description Class Sensor, Signature, *Timed Function usedValue*

PD/NF6 in
Description Class HallwaySection, ReqSpec, *Property* P_HS_3
Description Class HallwaySection, Signature, *Timed Predicate hwLightUncontroll*

PD/NF7 in
Main Description Class Floor, ReqSpec, *Property* P_F_9

PD/NF8 in
Main Description Class Floor, ReqSpec, *Property* P_F_{10}

PD/NF9 in
Description Class LightCtrlPanelRoom, ReqSpec, *Property* P_LCP_1
Description Class TemperatureCtrlPanel, ReqSpec, *Property* P_TCP_1

PD/NFH1 in
Description Class MalfunctionOccupancy, Signature, *Object manualEnteredOcc*
Description Class MalfunctionOccupancy, Signature, *Timed Predicate stateManualEnteredOcc*
Description Class MalfunctionSensor, ReqSpec, *Property* P_MS_1
Description Class MalfunctionSensor, Signature, *Object manualEnteredVal*
Description Class MalfunctionSensor, Signature, *Timed Function automaticDetermValue*
Description Class MalfunctionSensor, Signature, *Timed Function valueMal*
Description Class MalfunctionSensor, Signature, *Timed Predicate malDetEnv*
Description Class MalfunctionSensor, Signature, *Timed Predicate malDetMach*
Description Class MalfunctionSensor, Signature, *Timed Predicate stateManualEnteredValue*
Description Class Sensor, Signature, *Timed Function usedValue*

PD/NFH2 in

Main Description Class Floor, ReqSpec, *Property* P_F_{12}

PD/NFH3 in
Description Class MalfunctionSensor, Signature, *Timed Function automaticDetermValue*
Description Class MalfunctionSensor, Signature, *Timed Function valueMal*
Description Class MalfunctionSensor, Signature, *Timed Predicate malDetEnv*
Description Class MalfunctionSensor, Signature, *Timed Predicate malDetMach*
Description Class Room, ReqSpec, *Property* P_R_1
Description Class Room, ReqSpec, *Property* P_R_2
Description Class Room, ReqSpec, *Property* P_R_3
Description Class Room, ReqSpec, *Property* P_R_5
Description Class Room, ReqSpec, *Property* P_R_7
Description Class Room, Signature, *Timed Function malValvePos*
Description Class Room, Signature, *Timed Function tempRadMalSet*
Description Class Room, Signature, *Timed Predicate malTemp*
Description Class Room, Signature, *Timed Predicate malfunctionAllTempSens*
Description Class Sensor, Signature, *Timed Function usedValue*

PD/NFH4 in
Description Class EnteredValue, Signature, *Timed Function entered*
Description Class MalfunctionOccupancy, ReqSpec, *Property* P_MO_1
Description Class MalfunctionOccupancy, Signature, *Object manualEnteredOcc*
Description Class MalfunctionOccupancy, Signature, *Timed Predicate automaticDetermValue*
Description Class MalfunctionOccupancy, Signature, *Timed Predicate malOccDetEnv*
Description Class MalfunctionOccupancy, Signature, *Timed Predicate malOccDetMach*
Description Class MalfunctionOccupancy, Signature, *Timed Predicate occMal*
Description Class MalfunctionOccupancy, Signature, *Timed Predicate stateManualEnteredOcc*
Description Class MalfunctionSensor, Signature, *Timed Predicate malDetEnv*
Description Class MalfunctionSensor, Signature, *Timed Predicate malDetMach*
Description Class Occupancy, Signature, *Timed Predicate occUsed*
Description Class RoomTemperature, ReqSpec, *Property* P_RT_{10}
Description Class RoomTemperature, ReqSpec, *Property* P_RT_{11}
Description Class RoomTemperature, ReqSpec, *Property* P_RT_9
Description Class RoomTemperature, Signature, *Predicate defaultTempOccDaytime*
Description Class RoomTemperature, Signature, *Predicate defaultTempOccNighttime*
Global Signature, *Timed Predicate daytime*

PD/NFH5 in
Main Description Class Floor, ReqSpec, *Property* P_F_9

PD/NFH6 in
Main Description Class Floor, ReqSpec, *Property* P_F_{10}

PD/NFH8 in
Main Description Class Floor, ReqSpec, *Property* P_F_{13}

PD/NFH9 in
Description Class MalfunctionSensor, ReqSpec, *Property* P_MS_1
Description Class MalfunctionSensor, Signature, *Object manualEnteredVal*
Description Class MalfunctionSensor, Signature, *Timed Function automaticDetermValue*
Description Class MalfunctionSensor, Signature, *Timed Function valueMal*
Description Class MalfunctionSensor, Signature, *Timed Predicate stateManualEnteredValue*
Description Class Room, ReqSpec, *Property* P_R_7
Description Class Room, Signature, *Timed Function malValvePos*
Description Class Room, Signature, *Timed Function tempRadMalSet*
Description Class Sensor, Signature, *Timed Function usedValue*

PD/Paragraph 3 in

Description Class RoomLight, Signature, *Time Constant* T_RL_1
Description Class RoomLight, Signature, *Time Constant* T_RL_2
Description Class RoomLight, Signature, *Timed Function curLightScene*

PD/U4 in
Description Class Occupancy, Signature, *Timed Predicate occUsed*
Description Class RoomLight, ReqSpec, *Property* P_RL_{14}
Description Class RoomLight, ReqSpec, *Property* P_RL_8
Description Class RoomLight, Signature, *Time Constant* T_RL_1
Description Class RoomLight, Signature, *Time Constant* T_RL_2
Description Class RoomLight, Signature, *Timed Function curLightScene*
Description Class RoomLight, Signature, *Timed Function noFaultStandLightScene*

PD/U6 in
Description Class LightCtrlPanelRoom, ReqSpec, *Property* P_LCP_2
Description Class LightCtrlPanelRoom, Signature, *Object chosenLightScene*
Description Class RoomLight, ReqSpec, *Property* P_RL_{15}
Description Class RoomLight, ReqSpec, *Property* P_RL_9

PD/U7 in
Description Class LightCtrlPanelRoom, Signature, *Object ambLightLev*

PD/U8 in
Description Class RoomLight, ReqSpec, *Property* P_RL_4

PD/U9 in
Description Class RoomLight, ReqSpec, *Property* P_RL_5
Description Class RoomLight, Signature, *Timed Function curAmbLightLev*
Description Class RoomLight, Signature, *Timed Function defaultAmbLightLev*

PD/UC1 in
Description Class BlindedRoom, ReqSpec, *Property* P_BR_3
Description Class BlindedRoom, ReqSpec, *Property* P_BR_4

PD/UC2 in
Description Class BlindedRoom, ReqSpec, *Property* P_BR_4
Description Class BlindedRoom, Signature, *Timed Predicate manualBlind*

PD/UH1 in
Description Class ControlledInput, ReqSpec, *Property* P_CIN_2
Description Class ControlledInput, Signature, *Object allowedRange*
Description Class ControlledInput, Signature, *Object default*
Description Class ControlledInput, Signature, *Object userValue*
Description Class ControlledInput, Signature, *Timed Function value*
Description Class ControlledInput, Signature, *Timed Predicate userValueEntered*
Description Class EnteredValue, ReqSpec, *Property* P_EV_1
Description Class EnteredValue, Signature, *Timed Function checked*
Description Class EnteredValue, Signature, *Timed Function entered*
Description Class EnteredValue, Signature, *Timed Function nearestValue*
Description Class RoomTemperature, ReqSpec, *Property* P_RT_2
Description Class TempOperationModeSet, Signature, *Object ComfortMode*
Description Class TempOperationModeSet, Signature, *Timed Function roomDesiredTemperature*
Description Class TempOperationMode, Signature, *Timed Function desiredTemperature*
Description Class TemperatureCtrlPanel, Signature, *Object comfortTemperature*
Global Signature, *Domain COMFORT_TEMP*

PD/UH2 in
Description Class RoomTemperature, ReqSpec, *Property* P_RT_{12}
Description Class RoomTemperature, ReqSpec, *Property* P_RT_{13}

Description Class Room, Signature, *Timed Function tempRadNoMalSet*

PD/UH3 in
Description Class EnteredValue, Signature, *Timed Function entered*
Description Class Occupancy, Signature, *Timed Predicate occUsed*
Description Class TempOperationModeSet, Signature, *Object NonFreezeMode*
Description Class TempOperationModeSet, Signature, *Timed Predicate nonFreezeTime*
Description Class TemperatureCtrlPanelRoom, Signature, *Object nonFreezeOverride*

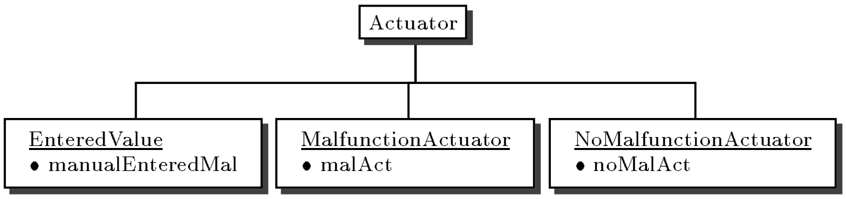
PD/UH4 in
Description Class ControlledInput, ReqSpec, *Property* P_CIN_2
Description Class ControlledInput, Signature, *Object allowedRange*
Description Class ControlledInput, Signature, *Object default*
Description Class ControlledInput, Signature, *Object userValue*
Description Class ControlledInput, Signature, *Timed Function value*
Description Class ControlledInput, Signature, *Timed Predicate userValueEntered*
Description Class DynamicInterval, Signature, *Object end*
Description Class DynamicInterval, Signature, *Object start*
Description Class EnteredValue, ReqSpec, *Property* P_EV_1
Description Class EnteredValue, Signature, *Timed Function checked*
Description Class EnteredValue, Signature, *Timed Function entered*
Description Class EnteredValue, Signature, *Timed Function nearestValue*
Description Class ExpectedUsage, Signature, *Object workingTime*
Description Class TempOperationModeSet, Signature, *Object OffMode*
Description Class TempOperationModeSet, Signature, *Object StandbyMode*
Global Signature, *Domain WORKING_TIME_RANGE*
Global Signature, *Timed Function now*

PD/UH5 in
Description Class CalendarInterval, Base Classes, *Class DynamicInterval*
Description Class CalendarInterval, Signature, *Timed Predicate actual*
Description Class ControlledInput, ReqSpec, *Property* P_CIN_2
Description Class ControlledInput, Signature, *Object allowedRange*
Description Class ControlledInput, Signature, *Object default*
Description Class ControlledInput, Signature, *Object userValue*
Description Class ControlledInput, Signature, *Timed Function value*
Description Class ControlledInput, Signature, *Timed Predicate userValueEntered*
Description Class DynamicInterval, Signature, *Object end*
Description Class DynamicInterval, Signature, *Object start*
Description Class EnteredValue, Signature, *Timed Function entered*
Description Class ExpectedUsage, Signature, *Object holidayTime*
Description Class TempOperationModeSet, Signature, *Object OffMode*
Global Signature, *Domain HOLIDAY_TIME_RANGE*
Global Signature, *Timed Function now*

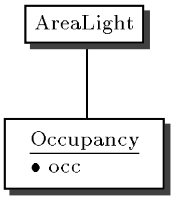
PD/UH6 in
Description Class BlindedRoom, ReqSpec, *Property* P_BR_3
Description Class ControlledInput, Signature, *Object userValue*
Description Class ControlledInput, Signature, *Timed Predicate userValueEntered*
Description Class EnteredValue, Signature, *Timed Function entered*
Description Class LightCtrlPanelRoom, Signature, *Object sunBlindSwitch*[

B Aggregation Hierarchies

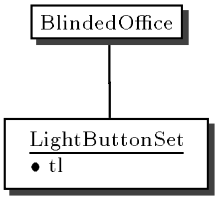
Aggregation Hierarchy of Description Class *Actuator*



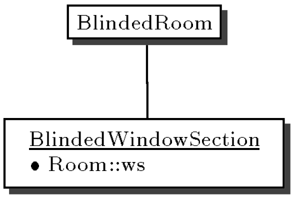
Aggregation Hierarchy of Description Class *AreaLight*



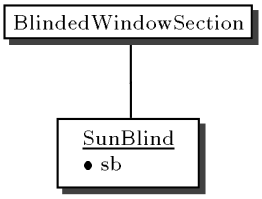
Aggregation Hierarchy of Description Class *BlindedOffice*



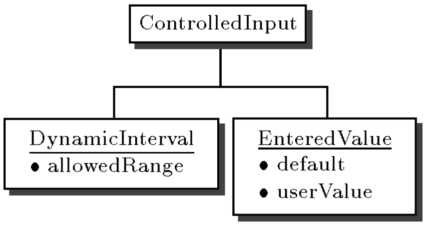
Aggregation Hierarchy of Description Class *BlindedRoom*



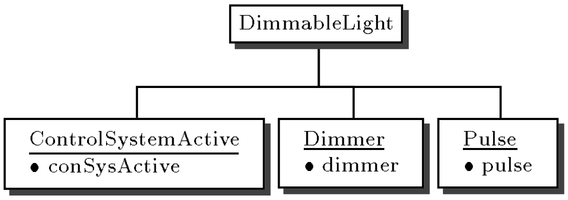
Aggregation Hierarchy of Description Class *BlindedWindowSection*



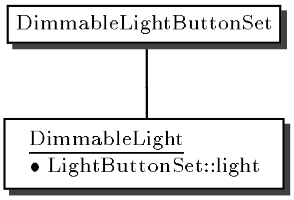
Aggregation Hierarchy of Description Class *ControlledInput*



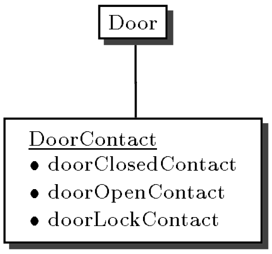
Aggregation Hierarchy of Description Class *DimmableLight*



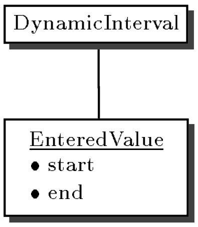
Aggregation Hierarchy of Description Class *DimmableLightButtonSet*



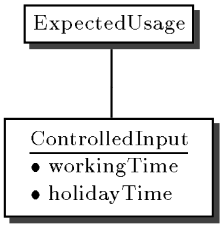
Aggregation Hierarchy of Description Class *Door*



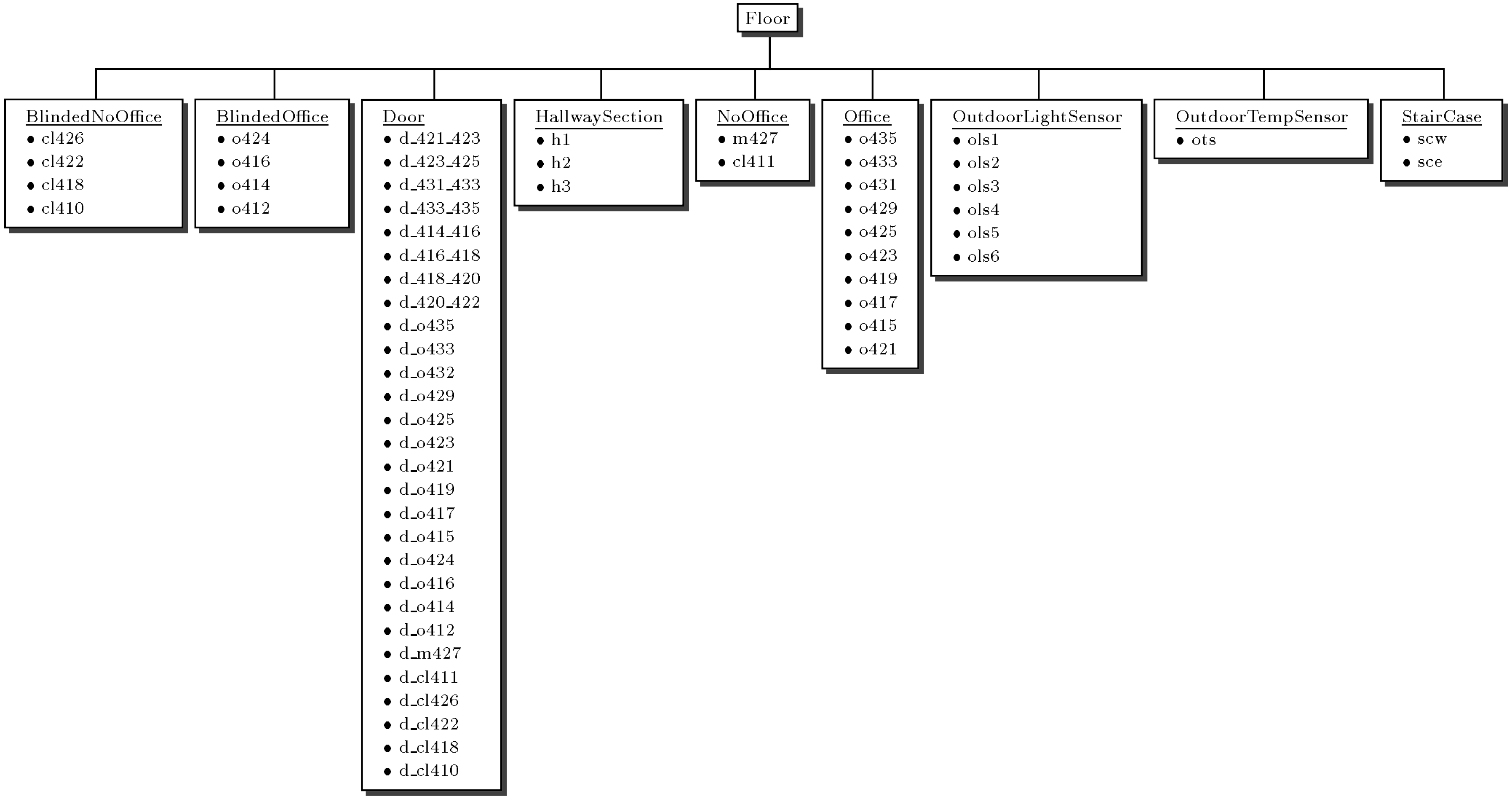
Aggregation Hierarchy of Description Class *DynamicInterval*



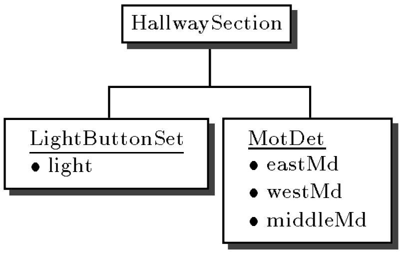
Aggregation Hierarchy of Description Class *ExpectedUsage*



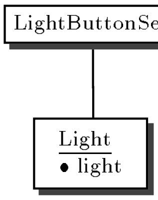
Aggregation Hierarchy of Description Class *Floor*



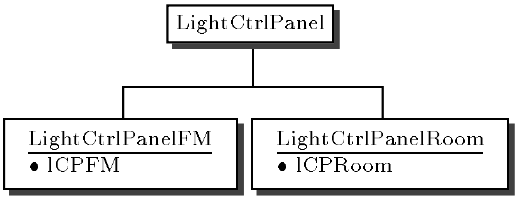
Aggregation Hierarchy of Description Class *HallwaySection*



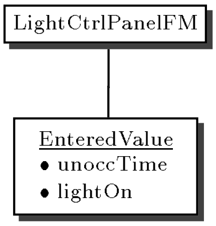
Aggregation Hierarchy of Description Class *LightButtonSet*



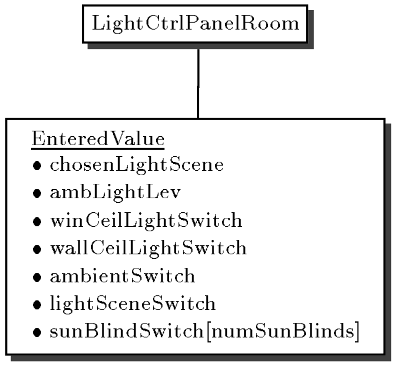
Aggregation Hierarchy of Description Class *LightCtrlPanel*



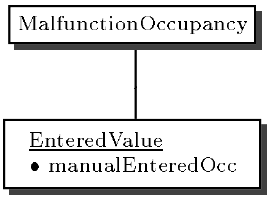
Aggregation Hierarchy of Description Class *LightCtrlPanelFM*



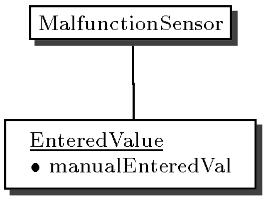
Aggregation Hierarchy of Description Class *LightCtrlPanelRoom*



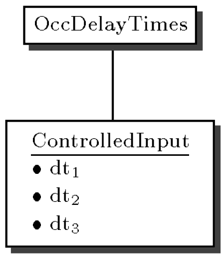
Aggregation Hierarchy of Description Class *MalfunctionOccupancy*



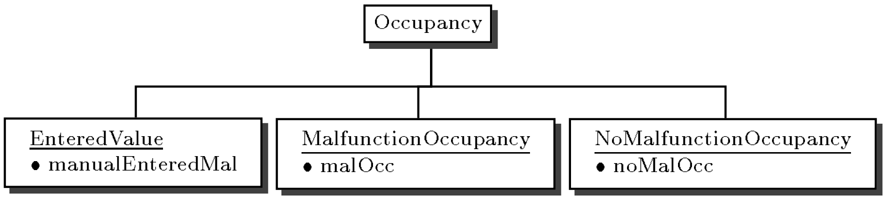
Aggregation Hierarchy of Description Class *MalfunctionSensor*



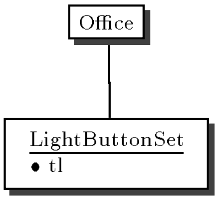
Aggregation Hierarchy of Description Class *OccDelayTimes*



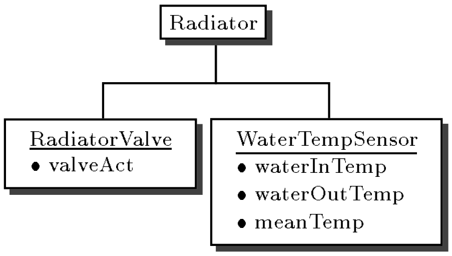
Aggregation Hierarchy of Description Class *Occupancy*



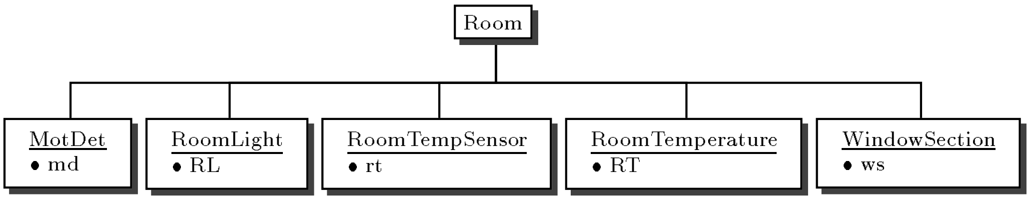
Aggregation Hierarchy of Description Class *Office*



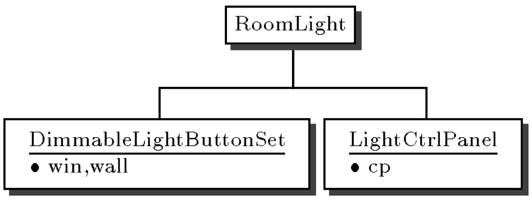
Aggregation Hierarchy of Description Class *Radiator*



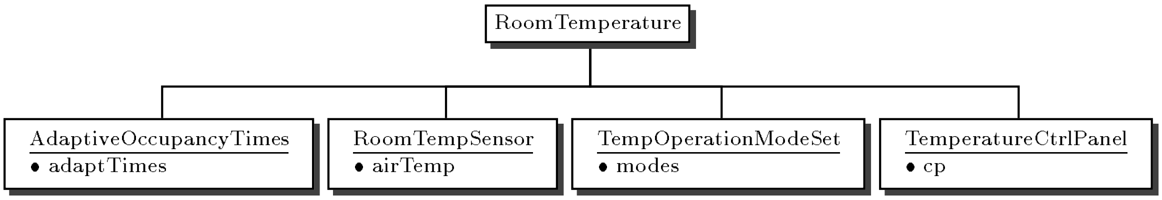
Aggregation Hierarchy of Description Class *Room*



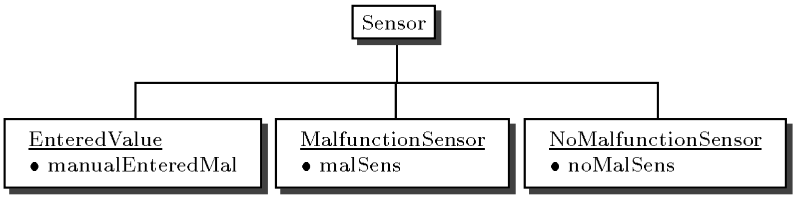
Aggregation Hierarchy of Description Class *RoomLight*



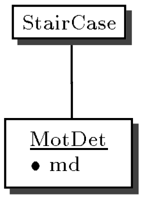
Aggregation Hierarchy of Description Class *RoomTemperature*



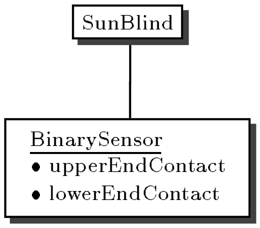
Aggregation Hierarchy of Description Class *Sensor*



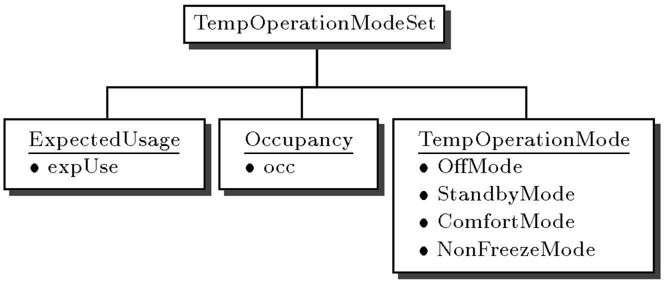
Aggregation Hierarchy of Description Class *StairCase*



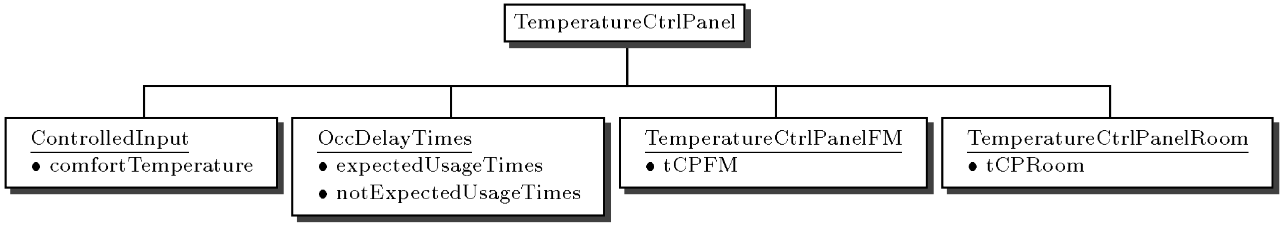
Aggregation Hierarchy of Description Class *SunBlind*



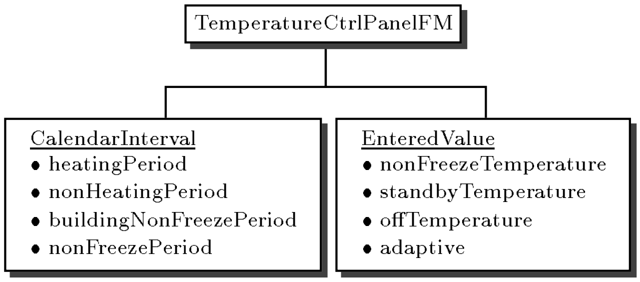
Aggregation Hierarchy of Description Class *TempOperationModeSet*



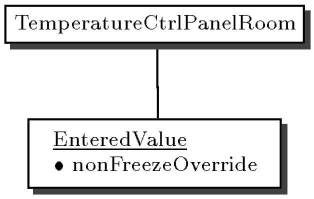
Aggregation Hierarchy of Description Class *TemperatureCtrlPanel*



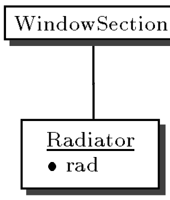
Aggregation Hierarchy of Description Class *TemperatureCtrlPanelFM*



Aggregation Hierarchy of Description Class *TemperatureCtrlPanelRoom*

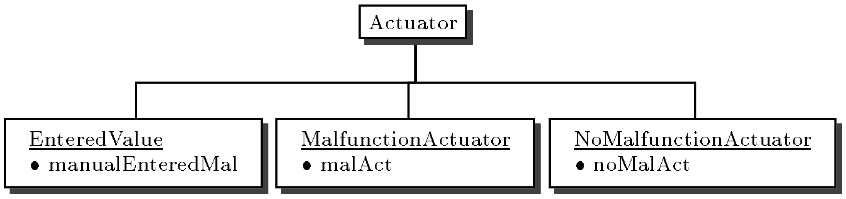


Aggregation Hierarchy of Description Class *WindowSection*

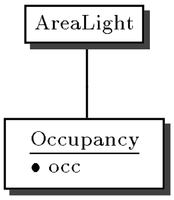


C Expanded Aggregation Hierarchies

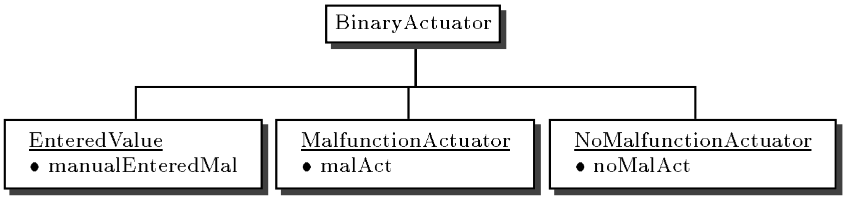
Expanded Aggregation Hierarchy of Description Class *Actuator*



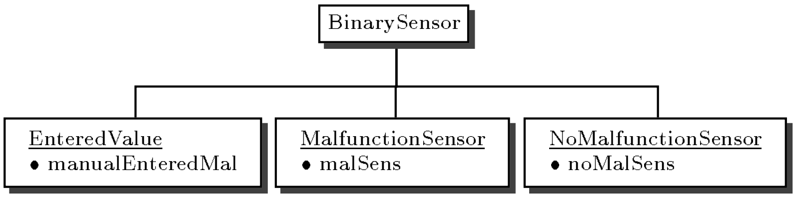
Expanded Aggregation Hierarchy of Description Class *AreaLight*



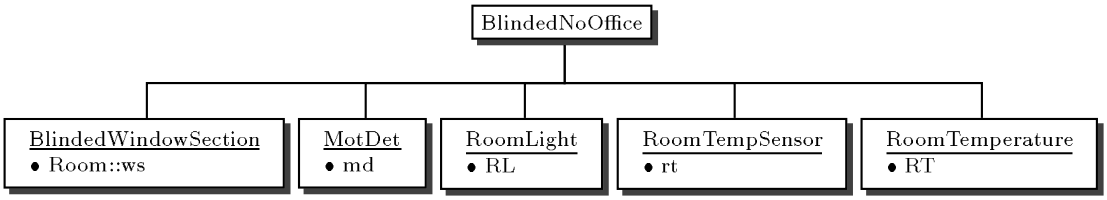
Expanded Aggregation Hierarchy of Description Class *BinaryActuator*



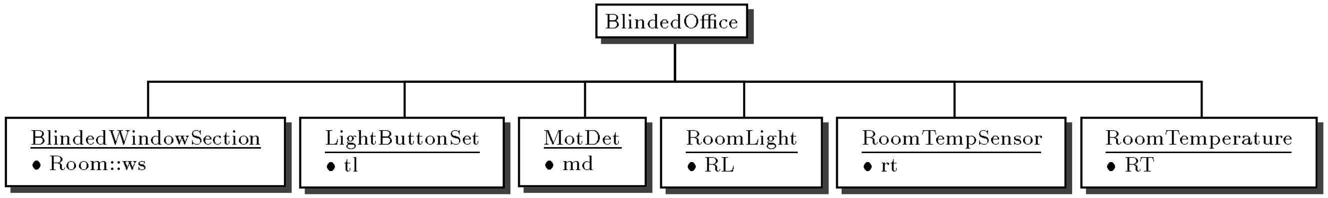
Expanded Aggregation Hierarchy of Description Class *BinarySensor*



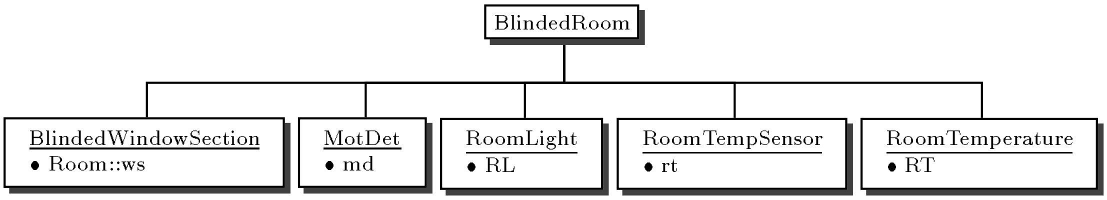
Expanded Aggregation Hierarchy of Description Class *BlindedNoOffice*



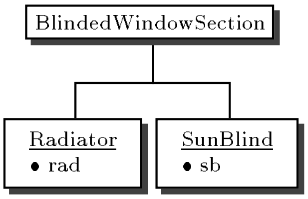
Expanded Aggregation Hierarchy of Description Class *BlindedOffice*



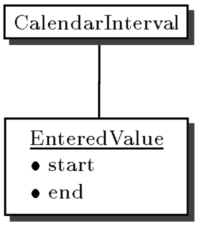
Expanded Aggregation Hierarchy of Description Class *BlindedRoom*



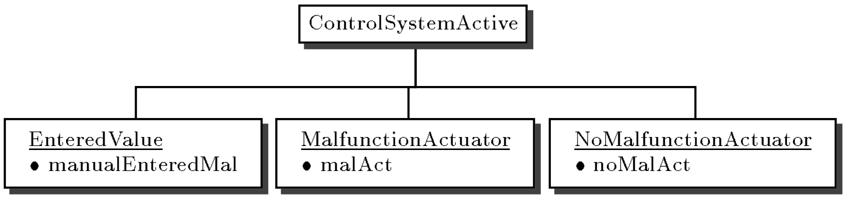
Expanded Aggregation Hierarchy of Description Class *BlindedWindowSection*



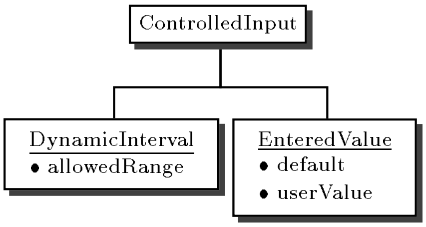
Expanded Aggregation Hierarchy of Description Class *CalendarInterval*



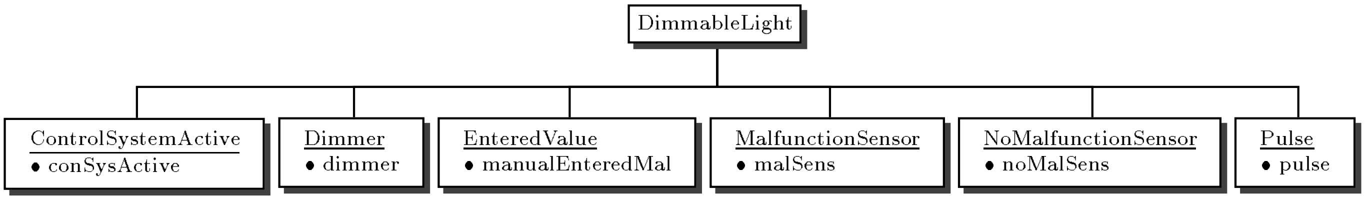
Expanded Aggregation Hierarchy of Description Class *ControlSystemActive*



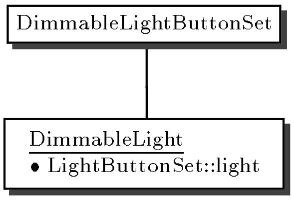
Expanded Aggregation Hierarchy of Description Class *ControlledInput*



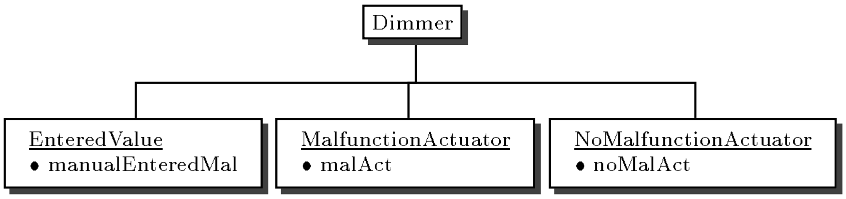
Expanded Aggregation Hierarchy of Description Class *DimmableLight*



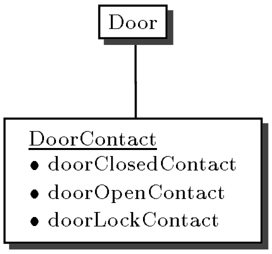
Expanded Aggregation Hierarchy of Description Class *DimmableLightButtonSet*



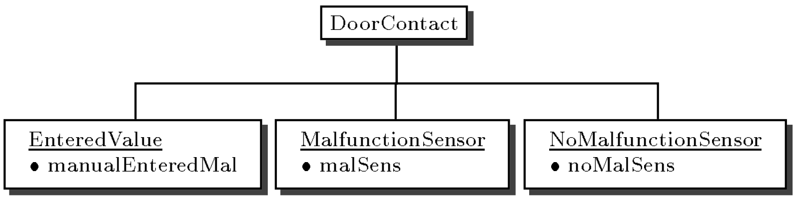
Expanded Aggregation Hierarchy of Description Class *Dimmer*



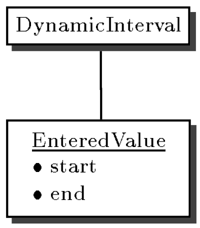
Expanded Aggregation Hierarchy of Description Class *Door*



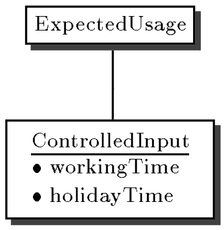
Expanded Aggregation Hierarchy of Description Class *DoorContact*



Expanded Aggregation Hierarchy of Description Class *DynamicInterval*



Expanded Aggregation Hierarchy of Description Class *ExpectedUsage*

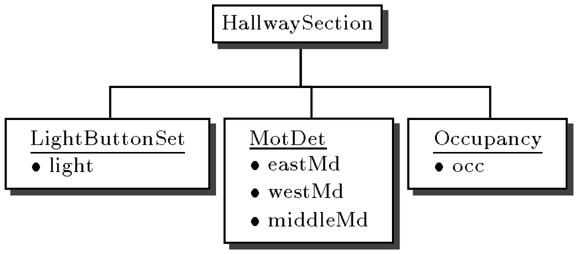


Expanded Aggregation Hierarchy of Description Class *Floor*

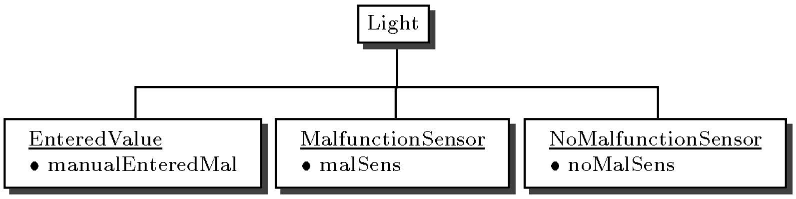
Floor

- BlindedNoOffice
  - cl426
  - cl422
  - cl418
  - cl410
- BlindedOffice
  - o424
  - o416
  - o414
  - o412
- Door
  - d_421_423
  - d_423_425
  - d_431_433
  - d_433_435
  - d_414_416
  - d_416_418
  - d_418_420
  - d_420_422
  - d_o435
  - d_o433
  - d_o432
  - d_o429
  - d_o425
  - d_o423
  - d_o421
  - d_o419
  - d_o417
  - d_o415
  - d_o424
  - d_o416
  - d_o414
  - d_o412
  - d_m427
  - d_cl411
  - d_cl426
  - d_cl422
  - d_cl418
  - d_cl410
- HallwaySection
  - h1
  - h2
  - h3
- NoOffice
  - m427
  - cl411
- Office
  - o435
  - o433
  - o431
  - o429
  - o425
  - o423
  - o419
  - o417
  - o415
  - o421
- OutdoorLightSensor
  - ols1
  - ols2
  - ols3
  - ols4
  - ols5
  - ols6
- OutdoorTempSensor
  - ots
- StairCase
  - scw
  - sce

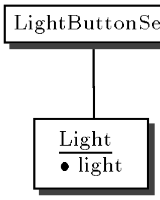
Expanded Aggregation Hierarchy of Description Class *HallwaySection*



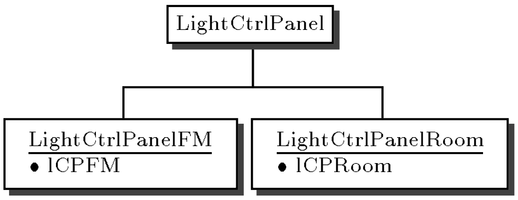
Expanded Aggregation Hierarchy of Description Class *Light*



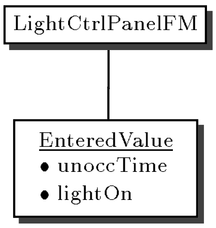
Expanded Aggregation Hierarchy of Description Class *LightButtonSet*



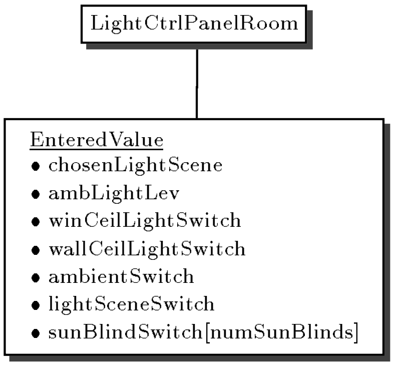
Expanded Aggregation Hierarchy of Description Class *LightCtrlPanel*



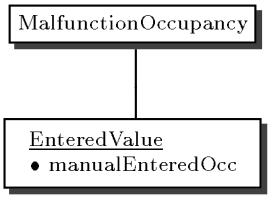
Expanded Aggregation Hierarchy of Description Class *LightCtrlPanelFM*



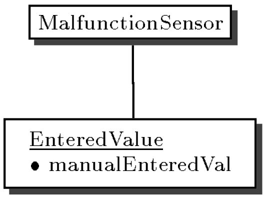
Expanded Aggregation Hierarchy of Description Class *LightCtrlPanelRoom*



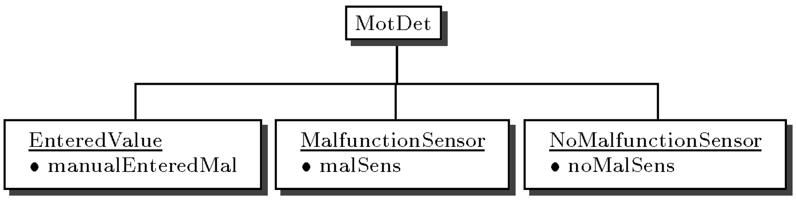
Expanded Aggregation Hierarchy of Description Class *MalfunctionOccupancy*



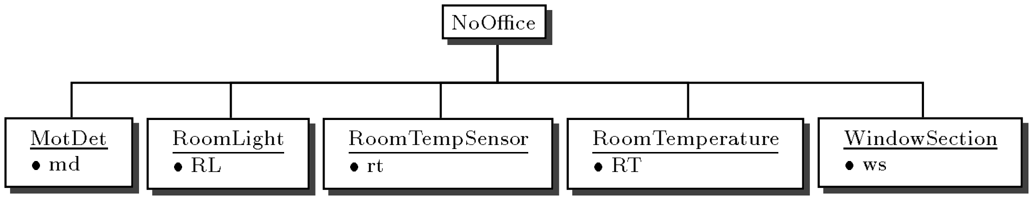
Expanded Aggregation Hierarchy of Description Class *MalfunctionSensor*



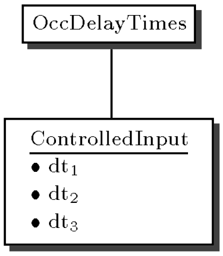
Expanded Aggregation Hierarchy of Description Class *MotDet*



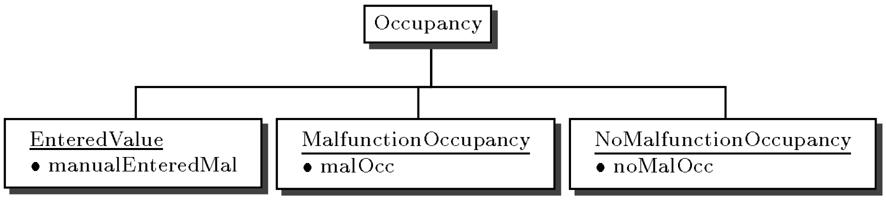
Expanded Aggregation Hierarchy of Description Class *NoOffice*



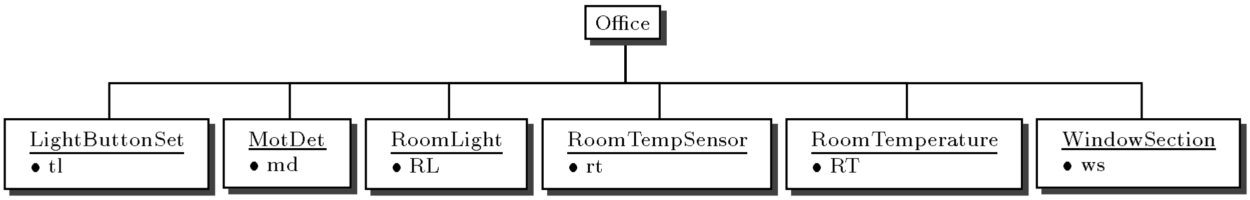
Expanded Aggregation Hierarchy of Description Class *OccDelayTimes*



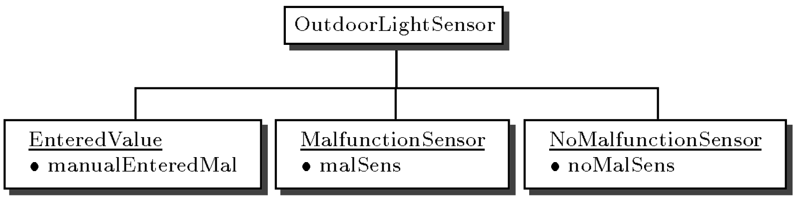
Expanded Aggregation Hierarchy of Description Class *Occupancy*



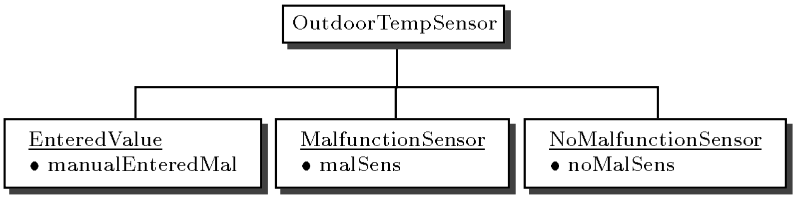
Expanded Aggregation Hierarchy of Description Class *Office*



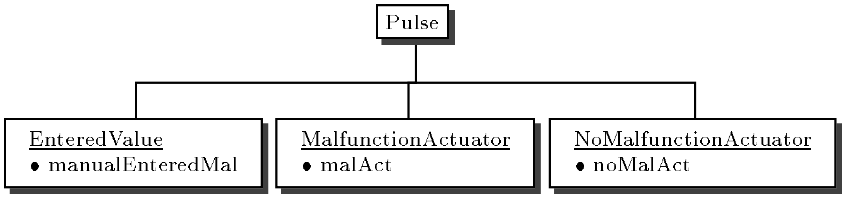
Expanded Aggregation Hierarchy of Description Class *OutdoorLightSensor*



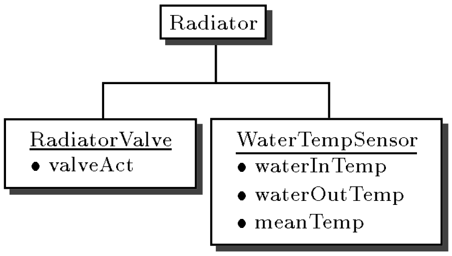
Expanded Aggregation Hierarchy of Description Class *OutdoorTempSensor*



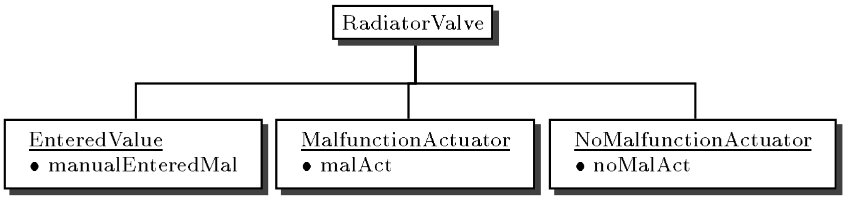
Expanded Aggregation Hierarchy of Description Class *Pulse*



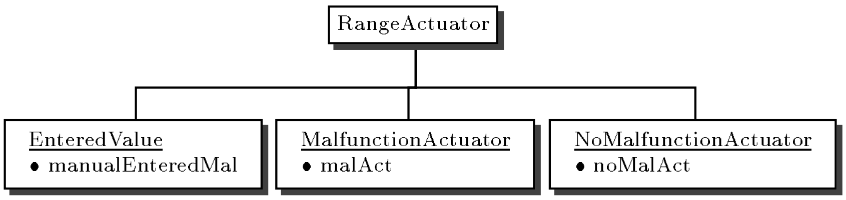
Expanded Aggregation Hierarchy of Description Class *Radiator*



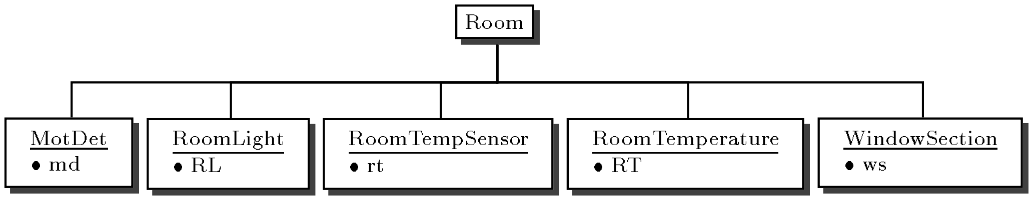
Expanded Aggregation Hierarchy of Description Class *RadiatorValve*



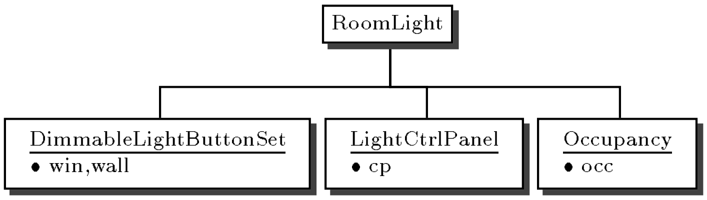
Expanded Aggregation Hierarchy of Description Class *RangeActuator*



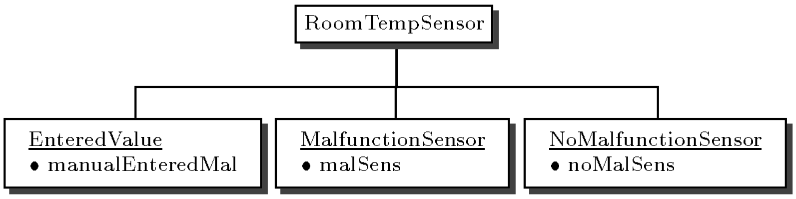
Expanded Aggregation Hierarchy of Description Class *Room*



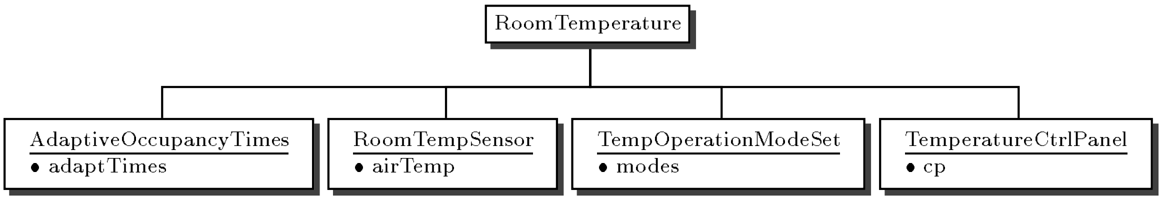
Expanded Aggregation Hierarchy of Description Class *RoomLight*



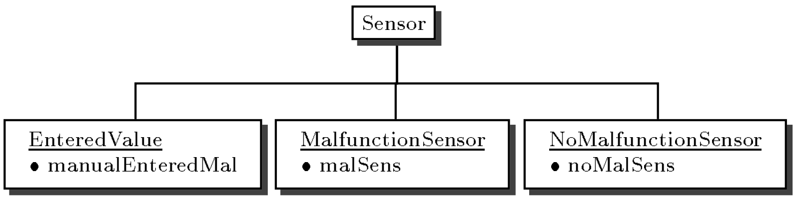
Expanded Aggregation Hierarchy of Description Class *RoomTempSensor*



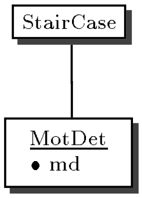
Expanded Aggregation Hierarchy of Description Class *RoomTemperature*



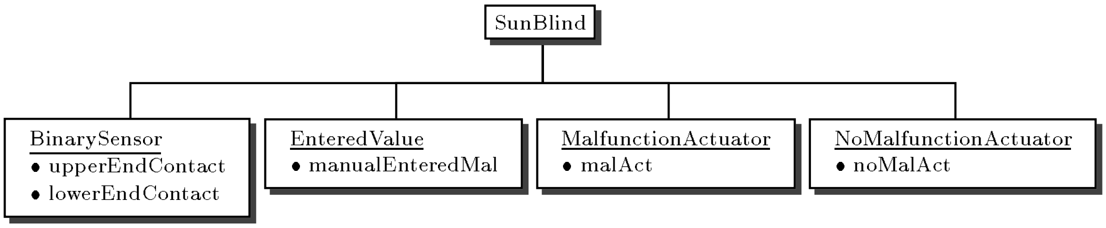
Expanded Aggregation Hierarchy of Description Class *Sensor*



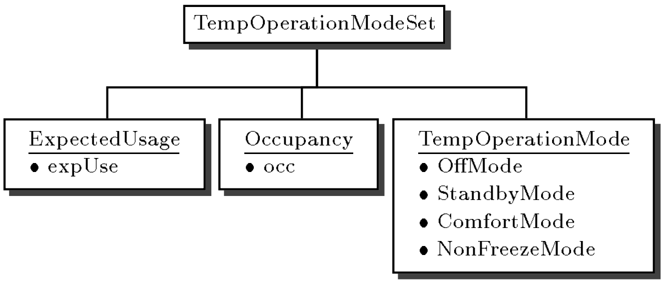
Expanded Aggregation Hierarchy of Description Class *StairCase*



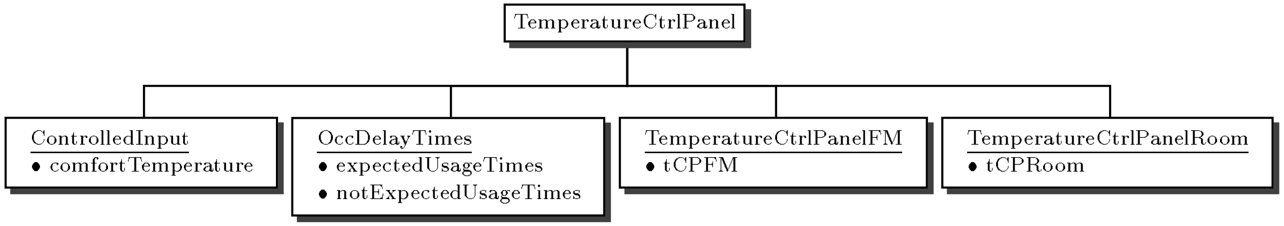
Expanded Aggregation Hierarchy of Description Class *SunBlind*



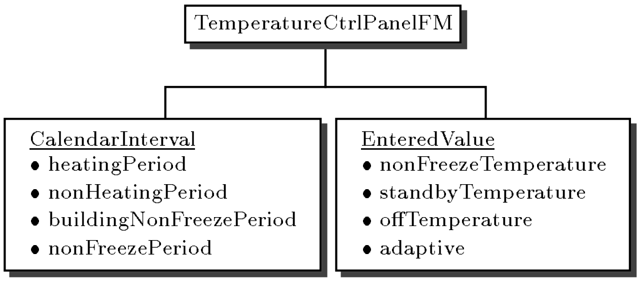
Expanded Aggregation Hierarchy of Description Class *TempOperationModeSet*



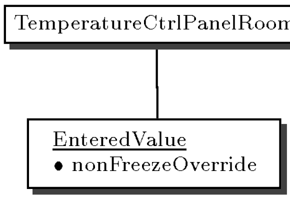
Expanded Aggregation Hierarchy of Description Class *TemperatureCtrlPanel*



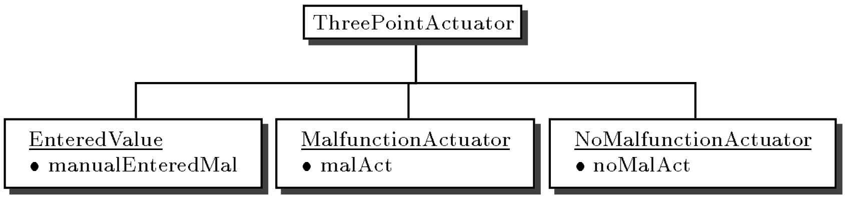
Expanded Aggregation Hierarchy of Description Class *TemperatureCtrlPanelFM*



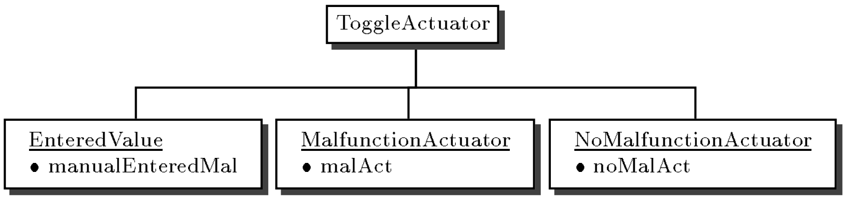
Expanded Aggregation Hierarchy of Description Class *TemperatureCtrlPanelRoom*



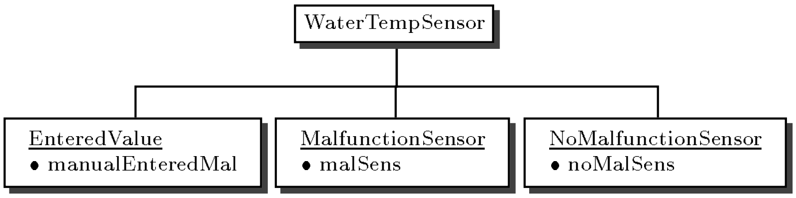
Expanded Aggregation Hierarchy of Description Class *ThreePointActuator*



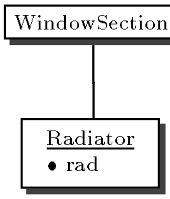
Expanded Aggregation Hierarchy of Description Class *ToggleActuator*



Expanded Aggregation Hierarchy of Description Class *WaterTempSensor*

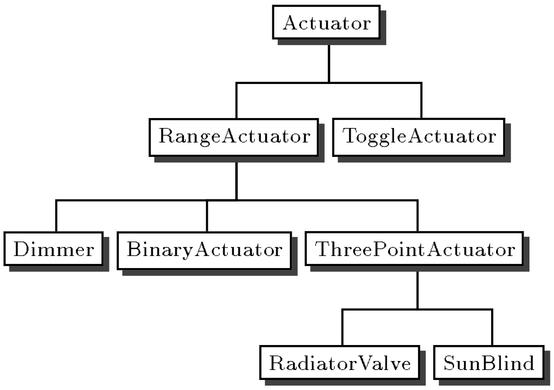


Expanded Aggregation Hierarchy of Description Class *WindowSection*

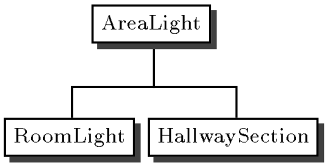


D Inheritance Hierarchies

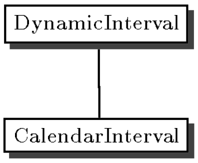
Inheritance Hierarchy of Actuator



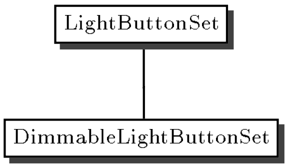
Inheritance Hierarchy of AreaLight



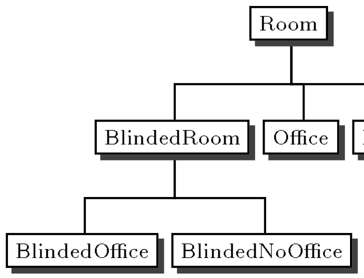
Inheritance Hierarchy of DynamicInterval



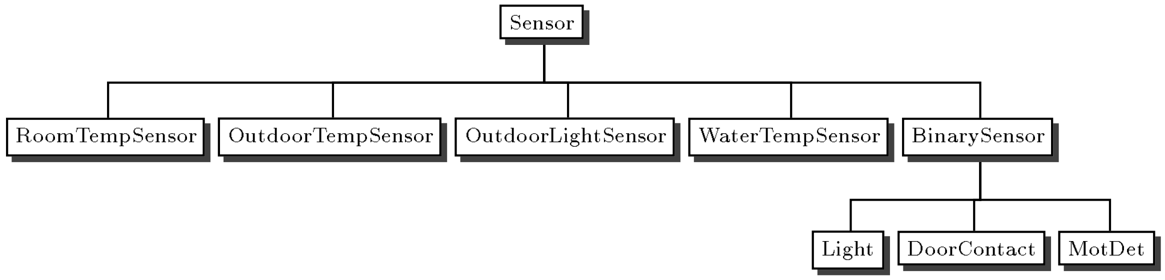
Inheritance Hierarchy of LightButtonSet



Inheritance Hierarchy of Room



Inheritance Hierarchy of Sensor



Inheritance Hierarchy of WindowSection