I. Troch, F. Breitenecker, eds. ISBN 978-3-901608-35-3

MODELLING OF THE REQUIREMENTS FOR DECENTRALIZED BUILDING AUTOMATION SYSTEMS

S. Runde, A. Fay
Helmut-Schmidt-University, Hamburg, Germany

Corresponding Author: S. Runde, Helmut-Schmidt-University Hamburg, Institute of Automation Technology
Holstenhofweg 85, 22043 Hamburg, Germany; stefan.runde@hsu-hh.de

Abstract. Modern building automation systems consist of up to 30000 devices (e.g. sensors, actuators, controller), which communicate via decentralized networks. Additionally, different sub-domains of the automation system, e.g. heating, ventilation, and lighting, are integrated in such a building automation system. Last but not least, within the Requirement-Elicitation phase engineering standards and aspects of energy efficiency are to be considered. Modelling of the requirements for these decentralized building automation systems, which are to be planned, results in a huge effort. In today's Requirement-Elicitation phase and Requirement Modelling, respectively, the requirements are merely written down in non-formalized, non-uniform text documents. This bears optimization potential for the engineering process in building automation. The whole planning process can benefit from a Requirement Model that covers all possible devices in the different sub-domains. Within this paper, the authors propose a solution for a formalized, uniformed Requirement Model for an improved engineering of building automation systems.

1 Introduction

Modern building automation systems (BAS) do not only provide improved comfort but offer significant energy cost savings, especially in office buildings and production halls, because of intelligent control systems such as intelligent lighting and sunblind functions. The assembly sections (AS) heating, ventilation, air-conditioning (HVAC), lighting, and shading (room automation (RA)) offer potential for huge energy savings, because of the considerable portion of energy consumption of buildings in these particular AS [1]. However, the initial investment costs in BAS are higher than the costs for conventional building installation. Thus, the long payback period of BAS scares many potential customers off the application of these systems. To achieve the possible energy savings, the initial investment costs need to be reduced.

Engineering costs account for a considerable part of the whole project cost of buildings. Consequently, the reduction of engineering costs can promote the dissemination of BAS. Looking at the building automation (BA) engineering process in general, optimization potential will become accessible by an automated process of design synthesis [2]. Within this automated process, automated engineering mechanisms can save costs [3], [4]. E.g. the ”AUTEG” (Automated Design for Building Automation) research project [5], [6], [7] focuses on this topic, and it is carried out by the Dresden University of Technology and the Helmut Schmidt University Hamburg along with industrial partners.

For the automated design synthesis, a uniform, formalized Requirement Model (RM) is mandatory for each sub-domain. It allows to represent the necessary requirements of a BAS, which is to be planned, for applications in further engineering phases.

Section 2 gives a short introduction into the state-of-the-art of designing BAS. In Section 3, a possible solution for the RM for decentralized BAS is described. The RM is described in Section 4. Finally, this paper gives a summary of the deliverables (cf. Section 5) and an outlook (cf. Section 6), which describes the motivation to represent the RM by means of the Semantic Web Technology OWL (Ontology Web Language) [8].

2 Engineering of Building Automation Systems –State-of-the-Art

In modern BA, decentralized automation networks gradually replace hierarchic automation network structures. Consequently, the importance of intelligent devices (e.g. sensors and actuators) increases for such automation networks and building automation systems, respectively [2]. These devices communicate in a ”peer to peer” way via a serial bus. These building BAS consist of up to 30000 different devices and are thus complex. In addition, many different manufacturers offer a large number of various devices each, and conformity is not guaranteed among devices of different manufacturers. Today, planners have to design such BAS manually, which requires huge effort. Besides this problem, they are confronted with the challenge of various assembly sections (AS), e.g. Heating, Ventilation, and Air Conditioning (HVAC), lighting and shading (room automation). Each device performs a certain task within an AS, and different tasks are often planned by different engineers. However, the high degree of linkage in modern BAS allows information interchange between the individual AS.

This typical state-of-the-art engineering process of BAS is an iterative process instead of an integrated design. Furthermore, a multitude of different planners and working groups are involved in several phases. The engineering process in BA can be subdivided into the phases *Requirement-Elicitation, Planning-Stage* and *Implementation-Planning*. In the following, the objectives in these several phases are outlined:

- Requirement-Elicitation: The intention of the requirement-elicitation phase is to elevate all relevant requirements for the two phases which will follow next. The results of the requirement-elicitation phase are non-formalized, non-uniformed “room books”.
- Planning-Stage: In the planning-stage, the requirements are processed, and a control structure is designed. The results are presented e.g. in control schematics or functions lists [9].
- Implementation-Planning: In this phase, specific components from specific manufacturers are selected based on the requirements of the requirement-eilicitation phase and e.g. in control schematics or functions lists of the planning-stage phase. The selected components are listed in text documents.

The main focus of this paper is on the Requirement-Elicitation phase. Today’s elicitation of requirements for the BAS, which is to be planned, depends on customers’ requests (e.g. a building owner) and on planner knowledge. This procedure often leads to sub-optimal requirements for the further engineering phases Planning-Stage and Implementation-Planning. Fundamental causes for this non-optimal requirements elicitation are e.g. that the building-owner and planner use different vocabularies, leading to misunderstandings, and that the planner cannot cope with all the functionalities and different types of components and devices respectively. Noticeable, there are only non-formalized, non-uniformed “room books” to describe the requirements to a BAS, which is to be planned.

But consistent requirements are important for both following engineering phases. Especially, flaws in early phases of the engineering workflow lead to increased costs in later phases and sub-optimal designed BAS. A uniform, formalized RM to describe the requirements of a decentralized BAS, which is to be planned, is mandatory for an improved Requirement-Elicitation and decreasing costs. Such a model allows defining consistent described requirements. In the next section, such a RM for the engineering of decentralized building automation systems is described.

3 Requirement Model

3.1 Class Structure

Within the AUTEG-Project [5], [6], [7], a catalogue of *Model Classes* and *Model Sub-Classes* for the Requirement-Elicitation has been gathered. This catalogue is the result of a detailed domain analysis, as based on, among others, [9], [11], [12] and VDI 3813-2 (which is in draft, and will be an inherent part of the ISO 16484-4 [13]). Thereby, the Model Classes are subsumed to the three levels *Basic Classes*, *Consumer Classes*, and *Continuous Classes* as shown in **Figure 1**.

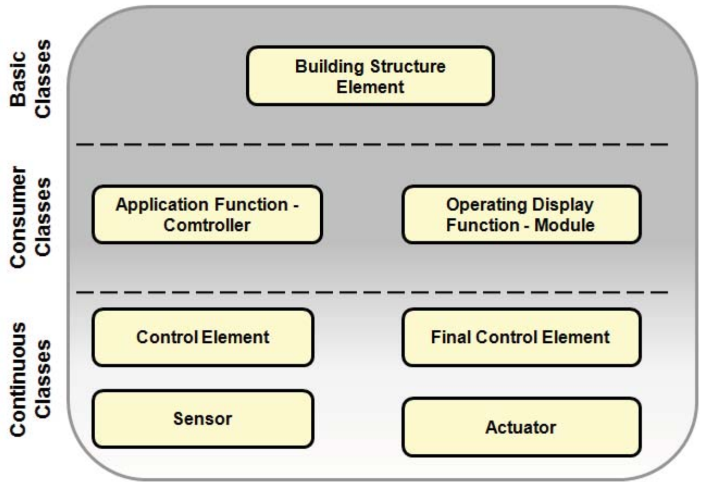


Figure 1. Model Classes

By means of the classes at the *Basic Class Level*, fundamental planning data can be modeled. This is the building structure, which is not planned by the BA-planner but by the architect. These requirements are necessary for the Requirement-Elicitation phase. The classes of the *Consumer Class Level* serve the modelling of consumer artifacts, however, the classes of the *Continuous Class Level* are primarily planner artifacts. In the following examples of *Model Sub-Classes* of the Basic Model Classes are outlined. These Sub-Classes reflect the components, devices and possible requirements respectively of a BAS, which is to be planned.

- Building Structure Element: *Building*, *Site, Space, Storey, etc.*
- Operating-/Display Function-Module: *Set Light-Module, Set Occupancy-Module, Set Scene-Module, etc.*
- Application Functions-Controller: *Constant Light Control-Controller, Temperature Control-Controller, etc.*

Operating-/Display Function-Modules and *Application Functions-Controllers* execute various types of functions. An Operating-/Display Function-Module represents the user interface primarily on a functional level. An Application Function-Controller can be perceived as a connector between sensors, final control elements, and control elements, as well as between Operating-/Display Function-Modules.

- Final Control Element: *Jalousie Drive, High/Low Voltage Halogen Lamp, Fluorescent Light, Incandescent Light, etc.*
- Actuator: *Lamp Actuator In-/ Direct, Jalousie Actuator In-/ Direct, etc.*
- Sensor: *Temperature Sensor, Occupancy Sensor, etc.*
- Control Element: *Switch, Control panel, etc.*

Besides the self-describing classes *Final Control Element, Actuator,* and *Sensor,* the class *Control Element* is defined. Control Elements share the characteristics of a final control element and sensor. They can engage into a physical process (actuator), but it is also possible to retrieve information from the physical process e.g. via human action (for example, a human operates a switch to rise the jalousie). The Operating-/Display Function-Modules permit the functional access to the BAS by means of the Control Elements.

The Model Sub-Classes are specified by *Device Attributes* (e.g. Mounting Form and Ingress Protection) and if necessary by *Functional Artifacts/Functions* (e.g. Actuate Light, Set Sunblind), and *Nonfunctional Artifacts/Hardware Modules* (e.g. Lamp Actuator Direct, Sunblind Actuator). The Functional and Nonfunctional Artifacts are again specified by *Artifact Attributes*. So, the Functional Artifact *Actuate Light* is specified by the attribute *Actuate Light Is Dimmable* for example. The Device Attributes and Artifact Attributes are described by several, typical *Default Values* with a *Unit*. So its is possible to outline the attribute *Operating Voltage* by the Default Values e.g. *12DC, 24DC* and the unit *V (Voltage).*

In **Figure 2**, the class model of the Model Classes is shown with the causal dependencies. For the purpose of an easier overall view, the Model Sub-Classes and their individual, causal dependencies (e.g. Constant Light Control-Controller - Illuminance Sensor) are not shown, as well as, the causal dependencies of every Model Class to the Building Structure Element class.

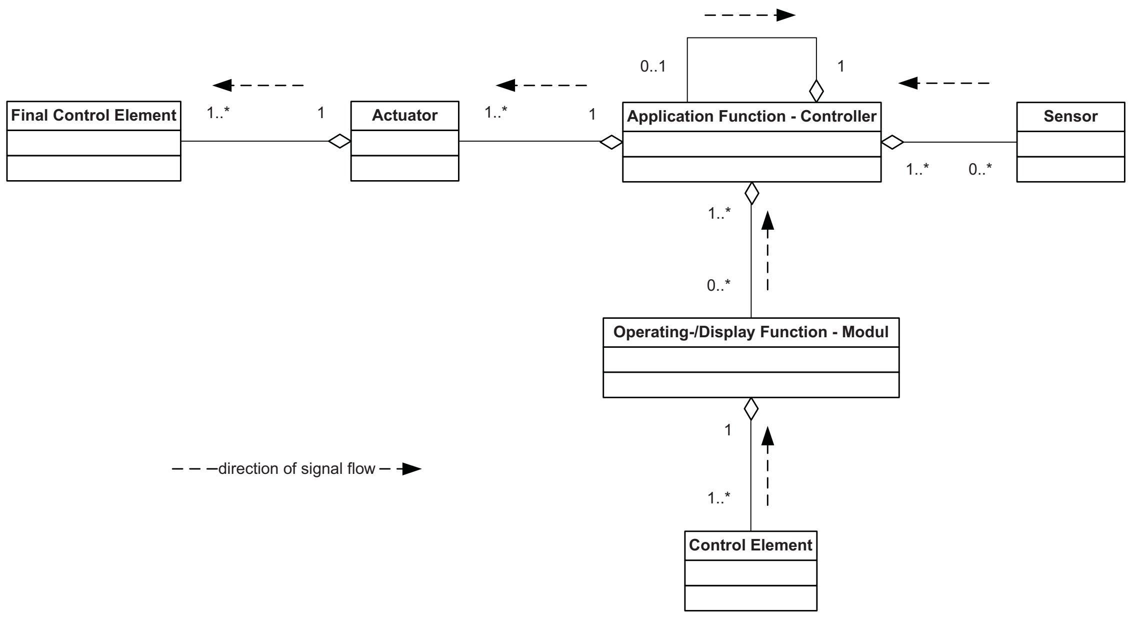


Figure 2. Class model of the Requirement Model with the direction of signal flow

The arrows (direction of signal flow) and the application function-controller causal dependency on itself are noticeable. This causality is intended to assign e.g. a scene controller to a constant light controller, which is a typical use case in practice. The signal flow is clarified in **Figure 4** in the next section.

3.2 Modelling example

An instance of the Lamp Actuator Direct class is shown in Figure 3. The figure illustrates especially the described facts regarding to Nonfunctional and Functional Artifacts (cf. Section 3.1).

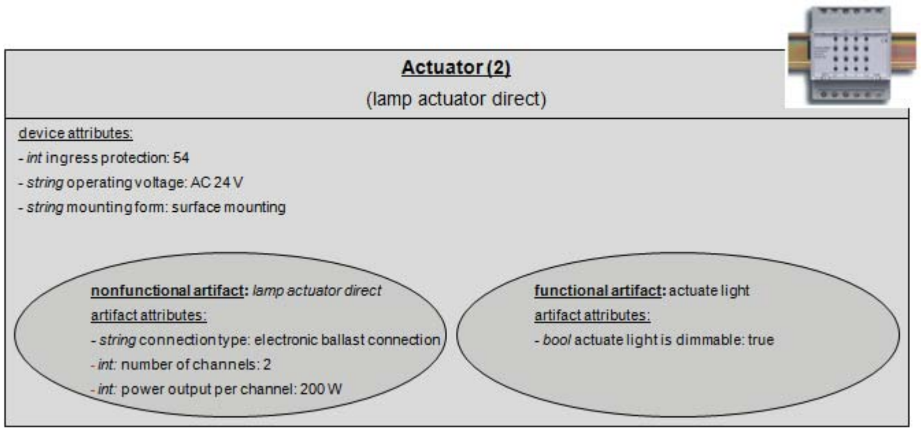

Actuator (2)
(lamp actuator direct)
device attributes:
- int ingress protection: 54
- string operating voltage: AC 24 V
- string mounting form: surface mounting
nonfunctional artifact: lamp actuator direct
artifact attributes:
- string connection type: electronic ballast connection
- int: number of channels: 2
- int: power output per channel: 200 W
functional artifact: actuate light
artifact attributes:
- bool actuate light is dimmable: true

Figure 3. Example for the model-element actuator

This instance Lamp Actuator Direct of the Model Sub-Class Actuator is characterized by the Device Attributes Ingress Protection, Operating Voltage, and Mounting Form, the Nonfunctional Artifact Lamp Actuator with its Artifact Attributes Connection Type, Number of Channels, and Output Per Channel and the Functional Artifact Actuate Light by the Artifact Attributes Actuate Light Is Dimmable. Additionally, all attributes are specified by a value and optionally unit.

In the **Figure 4**, room-requirements to a building automation system, which is to be planned, are modeled and instantiated respectively by means of the defined class structure (cf. Section 3.1). For the purpose of an easier overall view, all attributes, and the Functional and Nonfunctional Artifacts are not presented.

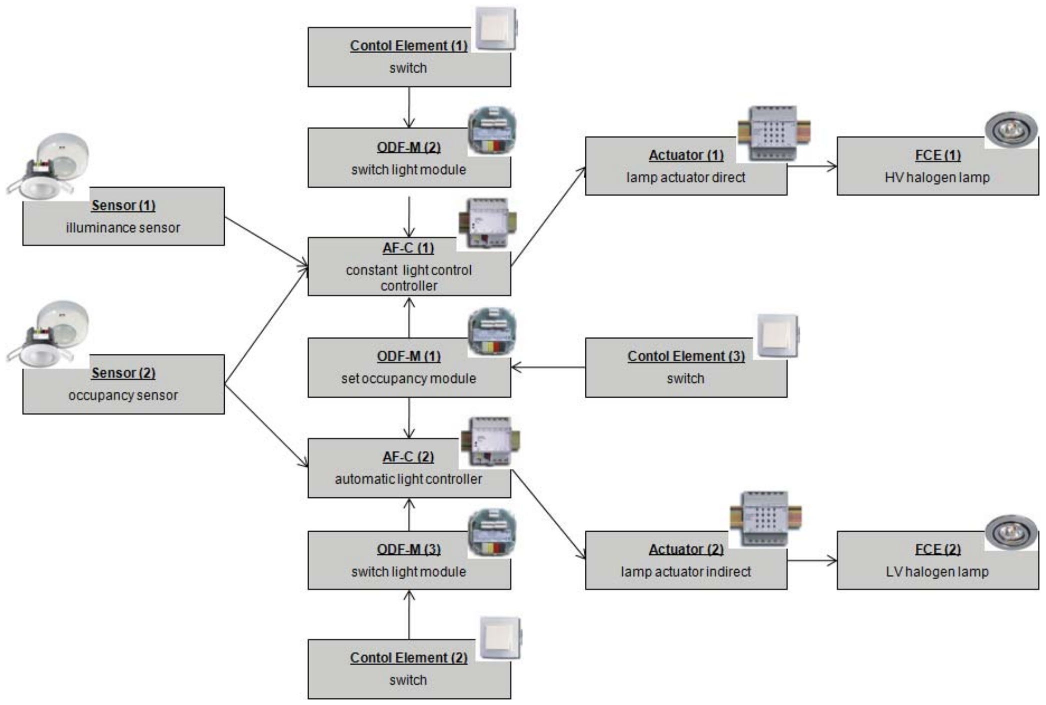


Figure 4. Requirement example for a room

The causal dependencies, which are modelled in the class structure are be considered by the instantiation of the Model Classes and Model-Sub-Classes. The signal flow (cf. Section) is also declared in Figure 4. For example, the class instances Application Function-Controllers *AF-C (1)* receives signals of the Operating-/Display Function-Modules *ODF-M (1)*, *ODF-M (2)* and *Sensor (1), Sensor (2)*. Furthermore, the *AF-C (1)* sends signals to *Actuator (1)*.

Beside the requirements for such a room the model allows to define *general attributes* (e.g. bus type, transmitting medium), which specifies requirements of a e.g. building or a floor. So, by means of a room-oriented procedure consistent requirements can be defined [7].

4 Data Exchange of the modelled requirements

In [14], data exchange formats e.g. IGES (Initial Graphics Exchange Specification) [15], STEP (Standard for the Exchange of Product Model Data) [16], IFC (Industry Foundation Classes) [17], and CAEX (Computer Aided Engineering eXchange) [18] are investigated. The intention is to fulfil an improved data exchange for an optimized engineering process in general and for the AUTEG-Project [5], [6], [7]. CAEX has been selected for the data exchange within the AUTEG-Project. CAEX is standardized in the IEC 62424 standard [18], and has been proved and tested in practice for process automation. CAEX is a meta model for vendor- and tool-independent structuring and categorization of CAE (Computer Aided Engineering) data. Nevertheless, it provides mechanisms to refer to vendor specific information. CAEX allows a systematic life-cycle-accompanying data exchange between the different CAE systems within the various AS in the different planning phases. CAEX does not define domain (e.g. process automation, building automation) or AS specific objects, because of its meta format concept. Based on CAEX, a Data Exchange Format for the Engineering of Building Automation Systems is designed and described in [14]. Because of the meta format basis, a specification of this Data Exchange Format to the terminology of the Requirement Modelling (Domain Model) is made. The basics of CAEX are shown in the following section.

4.1 CAEX basics

The four fundamental elements of CAEX are shown in Figure 5 and subsequently described shortly. Further information can be retrieved from [18].

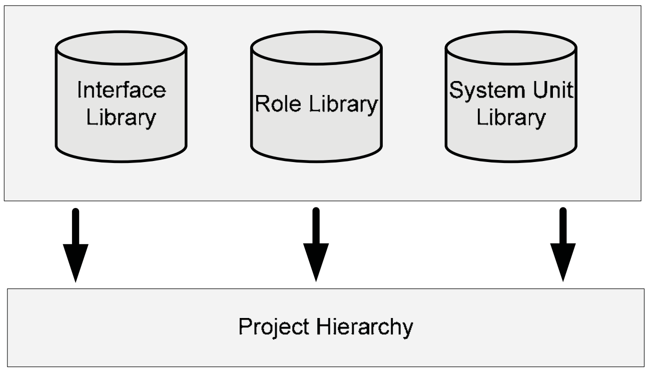


Figure 5. Basic elements of the CAEX meta model [10]

- In the *Interface Library*, possible interfaces for e.g. material flow, energy flow or information flow can be defined.
- By means of the *Role Library*, entities and functions can be described by symbolic placeholder (roles). Interfaces can be assigned to roles by referencing these to the interface library.
- Connected systems (units) consisting of functions and/or entities can be defined in the *System Unit Library* by referencing roles to the role library.
- In the *project hierarchy* the hierarchical structures with its entities and functions (including the automation systems) can be described. Therefore, roles of the role library and/or units of the unit library are to be referenced.

In principle, CAEX is independent of a special representation form, with which a CAEX conformant plant description is stored and exchanged between software programs. XML has been selected as the representation language for modelling with CAEX due to its readability and exchangeability [18], [19].

The following section describes briefly the specification of the domain model for building automation (BA) by means of CAEX regarding the requirement-modelling. CAEX can be seen as structuring guideline, because of its meta format concept. It imposes no restrictions on the content and organization of the data in this model for BA.

4.2 Specification of the Data Exchange Format

The specification of the four basic elements of CAEX (cf. Section 4.1) to the terminology of BA domain is shown in Figure 6.

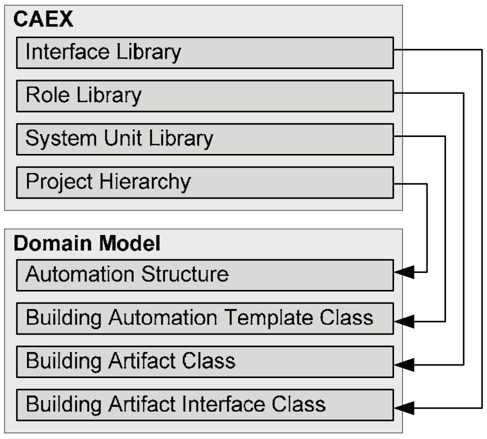
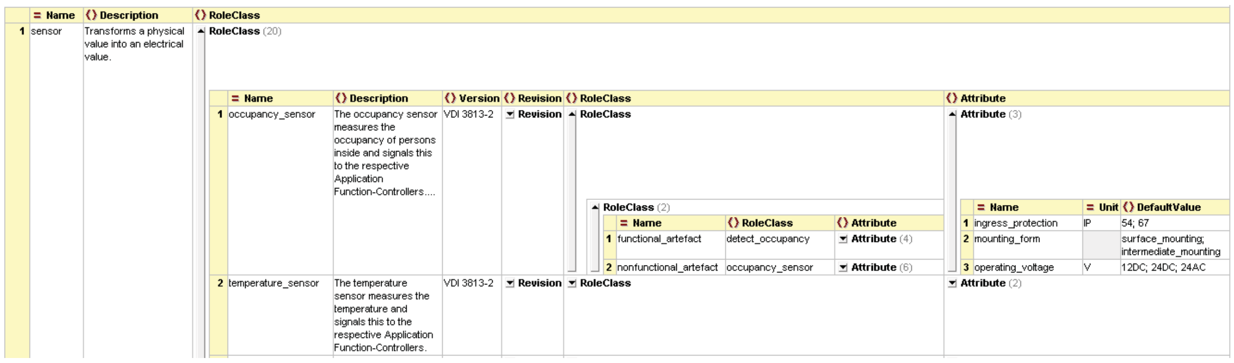


Figure 6. Specification of the four basic elements of CAEX to the BA domain

The Data Exchange Format for the Engineering of BAS [14] differentiates between the basic elements *Building Artifact Interface Class*, *Building Artifact Class*, *Building Automation Template Class*, and *Automation Structure*. In the following, these basic elements are briefly described regarding to the Requirement Modelling with the exeption of the Building Artifact Interface Class. This class is not used for the Requirement Modelling.

Building Artifact Class

Within the *Building Artifact Class*, the *Model Classes* (e.g. buidling structure element, application function-controller, sensor), and their *Model Sub-Classes* (constant-light control, illuminance sensor, etc.), the coresponding *Functional Artifacts* and *Nonfunctional Artifacts* are implemented. Furthermore, the *Device Attributes* and *Artifact Attributes* are implemented in this class. All these attributes are declared by several typical *Default Values* with a *unit* (cf. **Figure 7**). Additionally, all Model Classes are specified by a *Description*, a *Version* and a *Revision*. Furthermore, the causal dependencies are implemented in this Building Artifact Class (cf. Section 3.1), but not shown in **Figure 7**. This Figure presents details of the *Building Artifact Class*. For the purpose of easier reading, the following details of the domain model are shown in a graphic notation [20] and not in its textual XML notation.



	Name	Description	RoleClass
1	sensor	Transforms a physical value into an electrical value.	RoleClass (20)

RoleClass (20)

	Name	Description	Version	Revision	RoleClass	Attribute
1	occupancy_sensor	The occupancy sensor measures the occupancy of persons inside and signals this to the respective Application Function-Controllers....	VDI 3813-2	Revision	RoleClass	Attribute (3)
2	temperature_sensor	The temperature sensor measures the temperature and signals this to the respective Application Function-Controllers.	VDI 3813-2	Revision	RoleClass	Attribute (2)

RoleClass (2)

	Name	RoleClass	Attribute
1	functional_artefact	detect_occupancy	Attribute (4)
2	nonfunctional_artefact	occupancy_sensor	Attribute (6)

Attribute (3)

	Name	Unit	DefaultValue
1	ingress_protection	IP	54; 67
2	mounting_form		surface_mounting; intermediate_mounting
3	operating_voltage	V	12DC; 24DC; 24AC

Figure 7. Example of the Building Artefact Class

Building Artifact Template Class

The previously described Building Artifact Class is planner-specific and defines static model elements with the possibility of extension. The *Building Artifact Template Class* defines no specific elements, but the specification of the requirement model imposes restrictions on the structuring and categorization of the engineering data for the requirement-elicitation. By referencing artifacts of the *Building Artifact Class*, complex connected systems can be defined in this class. These planner-specific templates are "room-oriented" stored in this class. The room-oriented procedure facilitates the planning of building automation (BA), because these pre-defined connected systems (templates) yield a high degree of re-usability. To design these templates, the domain model and CAEX respectively provide mechanisms to reference (CAEX: `RefRoleClassPath`) and connect (CAEX: `InternalLink`) artifacts. Furthermore, the artifacts can be specified by additional *Attributes*. Thereby, the predefined *Default Values* of the Building Artifact Class can be used.

Automation Structure

In the *Automation Structure*, the requirements of a building automation system (BAS), which is to be planned, as an entire system can be described. This basic element serves as a basis for the hierarchical structure of a building. Therefore, the appropriate *Building Structure Elements* of the Building Artifacts class have to be instantiated. According to the corresponding Building Structure Element, Basic Model Sub-Classes of the Building Artifact Class can be instantiated in the hierarchical structure of a building to describe the BAS. In principle, for instantiation in CAEX the corresponding Basic Model Sub-Class has to be referenced. The pre-defined templates of the *Building Artifact Template Class* within the pre-defined references, attributes, etc. can also be instantiated according to the corresponding Building Structure Element. Naturally, it is possible to specify the instances with project specific *Descriptions*, *Versions, Revisions*, and *Attributes* with *Value* and *Unit.*

4.3 Data Exchange of Requirements by means of the Data Exchange Format

This section presents details of the requirement modelling within the AUTEG-Project. As an example of a building structure, the FZK-house (example building created by the IAI (International Alliance for Interoperability), [21]) was used (cf. **Figure 8**). Because of the FZK-house example does not provide the AS (e.g. HVAC, RA), the previous usual practice example (cf. **Figure 4**) is reused to explain the data exchange by means of this new RM.

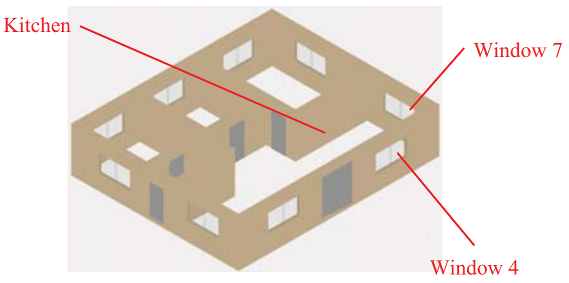
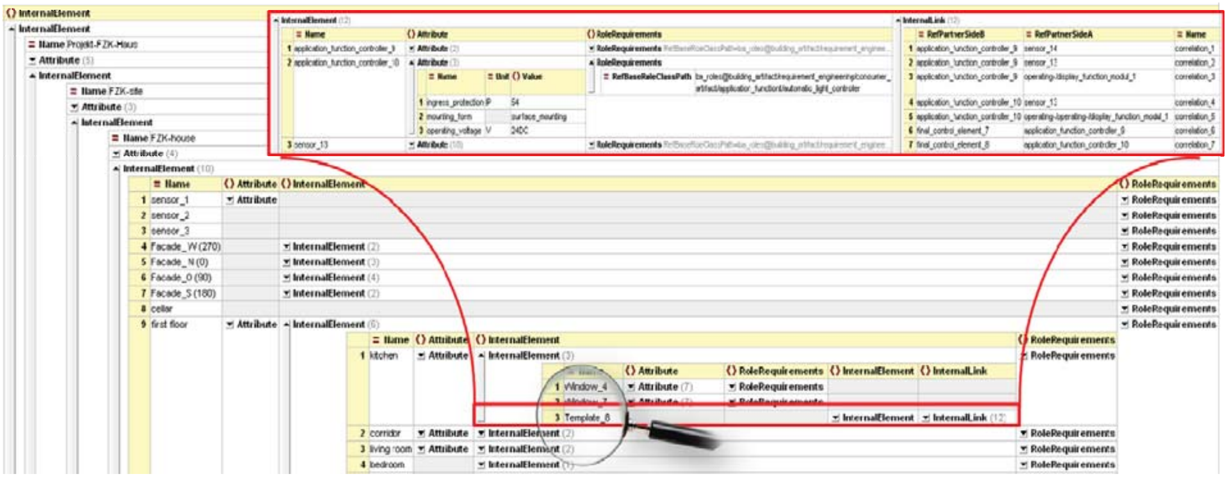


Figure 8. Detail of the FZK-house (first floor)

Figure 9 shows a detail of the *Automation Structure*. This detail represents the engineering data of the "Requirement-Elicitation" phase. The building structure of the FZK-house, with details of the first_floor, is shown. The `first_floor` is defined by the CAEX-element `InternalElement` and references to the corresponding Basic Model Sub-Class of the *Building Artifact Class* by the `RoleRequirements` element. Furthermore, the first_floor is specified by *Attributes*. Two windows (`window_7, window_4`) are defined within the kitchen. These windows use the `InternalElement` and are referred to by the `RoleRequirements` element (cf. **Figure 9**). The window assembly is an important information to a BA planner, e.g. for the HVAC assembly section (AS). This information renders it possible for a HVAC-planner to assign window-switches (which give a binary signal whether the window is opened or not) directly to a window, for example for a function to automatically stop the radiator in a room. Furthermore, the kitchen includes an instance of a predefined Template named `Template_8`, defined in the *Building Artifact Template Class* (cf. **Figure 9**/ magnification).



```
() InternalElement
 InternalElement
   Name Projekt-FZK-Haus
   Attribute (5)
   InternalElement
       Name FZK-site
       Attribute (3)
       InternalElement
           Name FZK-house
           Attribute (4)
           InternalElement (10)
```

	Name	() Attribute	() InternalElement	() RoleRequirements
1	sensor_1	Attribute		RoleRequirements
2	sensor_2			RoleRequirements
3	sensor_3			RoleRequirements
4	Facade_W (270)		InternalElement (2)	RoleRequirements
5	Facade_N (0)		InternalElement (3)	RoleRequirements
6	Facade_O (90)		InternalElement (4)	RoleRequirements
7	Facade_S (180)		InternalElement (2)	RoleRequirements
8	cellar			RoleRequirements
9	first floor	Attribute	InternalElement (6)	RoleRequirements

InternalElement (6) of first floor:

	Name	() Attribute	() InternalElement	() RoleRequirements
1	kitchen	Attribute	InternalElement (3)	RoleRequirements
2	corridor	Attribute	InternalElement (2)	RoleRequirements
3	living room	Attribute	InternalElement (2)	RoleRequirements
4	bedroom		InternalElement (1)	RoleRequirements

InternalElement (3) of kitchen:

	Name	() Attribute	() RoleRequirements	() InternalElement	() InternalLink
1	Window_4	Attribute (7)	RoleRequirements		
2	Window_7	Attribute (7)	RoleRequirements		
3	Template_8			InternalElement	InternalLink (12)

InternalElement (12):

	Name	() Attribute	() RoleRequirements
1	application_function_controller_9	Attribute (2)	RoleRequirements RefBaseRoleClassPath=ba_roles@building_artifact/requirement_enginee...
2	application_function_controller_10	Attribute (3)	RoleRequirements: RefBaseRoleClassPath ba_roles@building_artifact/requirement_engineering/consumer_artifact/application_function/automatic_light_controller
3	sensor_13	Attribute (10)	RoleRequirements RefBaseRoleClassPath=ba_roles@building_artifact/requirement_enginee...

Attribute (3) of application_function_controller_10:

	Name	Unit	() Value
1	ingress_protection	IP	54
2	mounting_form		surface_mounting
3	operating_voltage	V	24DC

InternalLink (12):

	RefPartnerSideB	RefPartnerSideA	Name
1	application_function_controller_9	sensor_14	correlation_1
2	application_function_controller_9	sensor_13	correlation_2
3	application_function_controller_9	operating-display_function_modul_1	correlation_3
4	application_function_controller_10	sensor_13	correlation_4
5	application_function_controller_10	operating-operating-display_function_modul_1	correlation_5
6	final_control_element_7	application_function_controller_9	correlation_6
7	final_control_element_8	application_function_controller_10	correlation_7

Figure 9. Details of the modelled FZK-house automation structure and an implemented template

The magnification shows explicit references (instances) to Basic Model Sub-Class of the *Building Artifact Class*. Various instances of artifacts are listed within this instantiated template_8, e.g. application_function_9, application_ function_10, sensor_13. In this case, application function 10 is an instance of the Basic Model Sub-Class `automatic light`. Furthermore, this instance is specified by the attribute e.g. ingress protection of 54 IP. The `InternalLinks` (CAEX-element) link the instances, e.g. various *Final Control Elements*, *Sensors*, *Control Elements, Application Function-Controllers,* to a complex BAS.

5 Conclusions

A main topic of this paper is the modelling of the requirements for decentralized building automation systems, which are to be planned. Thereto, the Basic Model Classes e.g. Application Function-Controllers, Sensors and Basic Model Sub-Classes (devices and possible requirements) e.g. Constant Light Control-Controller, Stairway Lighting-Controller, Illuminance Sensor are defined in Section 3.1. The Basic Model Sub-Classes are specified by Device Attributes, and if necessary by Functional Artifacts/Functions, and Nonfunctional Artifacts/Hardware Module. The Functional and Nonfunctional Artifacts are again specified by Artifact Attributes. Additionally, the possible correlations between these Sub-Classes (e.g. Constant Light Control-Controller "needs an" Illuminance Sensor) are presented. Closing this topic a short practice-example of modelling by using these requirement model elements is shown.

Another main topic of this paper is the data exchange of the modeled requirements by using the Data Exchange Format for the Engineering of Building Automation Systems [14]. This format is based on CAEX (Computer Aided Engineering eXchange) [10] [18], which is a meta model for vendor- and tool-independent structuring and categorization of CAE data and does not define domain specific patterns. Hence, a specification of CAEX, also referred to as modelling, to the terminology of the requirement-modelling is shown in this paper. The previous example is reused to explain the data exchange. The research activities within the AUTEG-Project show that the requirements represented by means of the data exchange format leads to an improved and more efficient BAS engineering.

6 Outlook

Finally, this paper describes the motivation to represent the RM by means of the Semantic Web Technology. (Ontology Web Language) [6] [22]. The description of the knowledge (cf. Section 3.1) in OWL offers, among other advantages, the following fundamental advantages:

- Formal representation of facts, which allows to use standard computing science technologies (e.g. Reasoner (e.g. Racer [24]), inference-machines (e.g. Jess [25]), Rules (e.g. SWRL (Semantic Web Rule Language [26])), Rule Exchange Formats (e.g. RuleML (Rule Markup Language [27])) for the knowledge based engineering in the context of automation.
- Using quasi-standards (e.g. OWL, SWRL) of the W3C (World Wide Web Consortium). The huge community of the W3C guarantees almost a permanent, further development.
- Using these standard computing science technologies allows more and more focussing on better methods for the application of engineering of automation systems. The computing science is the "tool" for a better engineering of automation systems, as well as for an automation of the engineering of automation systems.

In **Figure 10**, the possible "future" Knowledge-Based System for the engineering of BAS and Automation in general is proposed. The inference machine in **Figure 10** writes the inferenced results back to the fact base. But it is also possible to write these results to another format by means of the inference machine.

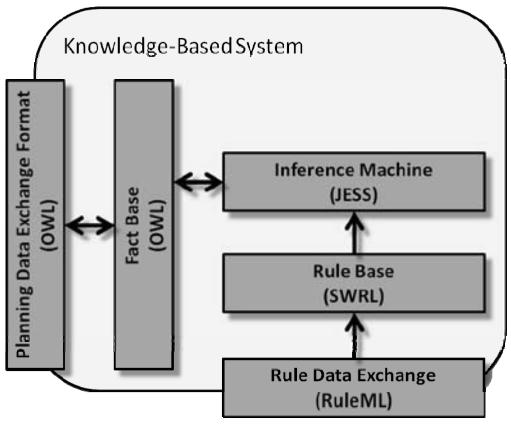


Figure 10. Knowledge-based system in context of engineering of automation systems using Semantic Web Technologies

Data Exchange between the engineering tools in the different planning phases and AS still happens by CAEX. The reason for this decision is on the one hand, that OWL is not a Mark-Up Language for data exchange, but it is a language to describe knowledge. On the other hand, the reason why CAEX is used as the data exchange format is, that CAEX is proved, tested and used in practice in the automation domain [14], [28], [29]. The concept and a prototyping implementation of the transformation of CAEX to OWL is described in [30].

7 References

[1] Kreider, J.-R.: *Handbook of heating, ventilation, and air conditioning*, Boca Raton, CRC Press, 2001

[2] Kabitzsch, K.: *Tools*, in: Loy, D.; Dietrich, D.; Schweinzer, H. (Eds.): *Open Control Networks*, Boston, Dordrecht, London, Kluwer Academic Publishers, pp.59-72, 2001

[3] Schmidberger, T.; Horch, A., Fay, A., Drath, R., *Rule Based Engineering Of Asset Management System Functionality*, in 5th Vienna Symposium on Mathematical Modelling, Vienna. Breitenecker, F.; I. Troch, pp. 2.1–2.8, 2006

[4] Drath, R.; Fay, A.; Schmidberger, T.: *Computer aided design and implementation of interlocking control code*, in 2006 IEEE International Symposium on Computer-Aided Control Systems Design (CACSD06), Munich, IEEE, pp. 2653-2658, 2006

[5] Dibowski, H.; Oezluek, C.; Ploennigs, J.; Kabitzsch, K.: *Realizing the Automated Design of Building Automation Systems*, in IEEE International Conference on Industrial Informatics (INDIN), 2006, pp. 251–256, Singapore.

[6] University of Technology Dresden: *Automated Design for Building Automation (AUTEG)*, University of Technology *Dresden*, 2007, http://www.ga-entwurf.de

[7] Runde, S.; Dibowski, H.; Fay, A.; Kabitzsch, K.: *Integrated Automated Design Approach of Building Automation Systems*, in proceedings of 2008 IEEE Emerging Technologies and Factory Automation (ETFA 2008), Hamburg, pp. 1280-1288, 2008.

[8] Ontology Web Language (OWL): http://www.w3.org/TR/owl-features/, 2004.

[9] International Organization for Standardization: *ISO 16484-3 – Buidling automation and control systems (BACS) - Part 3: Functions*

[10] M. Fedai and R. Draht, *CAEX - a neutral data exchange format for engineering data*, atp international, 2005.

[11] International Organization for Standardization: *ISO 16484-2—Building automation and control systems (BACS)—Part 2: Hardware*

[12] International Organization for Standardization: *ISO/PAS 16739: Industry Foundation Classes, Release 2x, Platform Specification (IFC2x Platform)*

[13] International Organization for Standardization: *ISO 16484-4—Building automation and control systems (BACS)—Part 4: Applications*

[14] Runde, S.; Fay, A.: *Data Exchange Format for the Engineering of Building Automation Systems*, in proceedings of 2008 IEEE Emerging Technologies and Factory Automation (ETFA 2008), Hamburg, pp. 303-310, 2008.

[15] National Institute of Standards and Technology: *ANS US PRO/IPO-100-1996 – Initial Graphics Exchange Specification*, http://www.uspro.org/documents/IGES5-3_forDownload.pdf

[16] International Organization for Standardization: *ISO 10303 – Industrial automation systems and integration – Product data representation and exchange*

[17] International Organization for Standardization: *ISO/PAS 16739: Industry Foundation Classes*, Release 2x, Platform Specification (IFC2x Platform)

[18] International Electrotechnical Commission*: IEC PAS 62424 - Representation of process control engineering requests in P&I diagrams and data exchange between P&ID tools and PCE-CAE tools*

[19] Fay, A.: *XML for the Exchange of Automation Project Information*, in The Industrial Information Technology Handbook, pp. 98.1–98.8. CRC Press, South San Francisco, 2005

[20] Altova, *Altova XML Spy*, 2005, http://www.altova.com

[21] National Institute of Standards and Technology: *ANS US PRO/IPO-100-1996 – Initial Graphics Exchange Specification*, http://www.uspro.org/documents/IGES5-3_forDownload.pdf

[22] W3C (Word Wide Web Consortium): *W3C Semantic Web Activity*, http://www.w3.org/2001/sw/, 2008

[23] W3C (Word Wide Web Consortium): *OWL Web Ontology Language Guide*, http://www.w3.org/TR/2004/REC-owl-guide-20040210; 2008

[24] Racer Systems: *Racer*, http://www.racer-systems.com/de/index.phtml, 2008

[25] Jess, the Rule Engine: *Jess*, http://www.jessrules.com/jess/index.shtml, 2008

[26] W3C: *SWRL: A Semantic Web Rule Language - Combining OWL and RuleML*, http://www.w3.org/Submission/SWRL/, 2008

[27] The Rule Markup Initiative: *RulemML*, http://www.ruleml.org/, 2008

[28] AutomationML Group: *AutomationML,* http://www.automationml.org/, 2008

[29] Güttel**,** K**.;** Weber, P.; Fay, A.: Automatic generation of PLC code beyond the nominal sequence. in proceedings of 2008 IEEE Emerging Technologies and Factory Automation (ETFA 2008), Hamburg, 15.-18. September 2008.

[30] Runde, S.; Güttel, K.; Fay, A.: *Transformation von CAEX-Anlagenplanungsdaten in OWL – Eine Anwendung von Semantic Web Technologien*, accepted for publication in proceedings "Automation 2009", Baden-Baden, 6.-7. Juni 2009.

Acknowledgements

The authors gratefully acknowledge the financial support provided by the German Federal Ministry of Economics and Technology along with industrial partners in the frame of the AUTEG project. (project reference IN-5506).