Switching to a holistic perspective on semantic component models in building automation – tapping the full potential of automated design approaches

Switching to a holistic perspective on semantic component models in building automation – tapping the full potential of automated design approaches

Wollschlaeger B, Eichenberg E and Kabitzsch K
Chair of Technical Information Systems, Technische Universität Dresden, 01062 Dresden, Germany

{bastian.wollschlaeger, elke.eichenberg, klaus.kabitzsch}@tu-dresden.de

Abstract. Building automation systems are vital for reducing both energy consumption and carbon footprint of modern buildings. However, the engineering of such building systems is becoming increasingly complex, such that automating the engineering tasks is inevitable. Therefore, semantic component models of high quality and high expressiveness are required. But, as of today, existing models suffer from focusing on isolated aspects, high effort for component model specification, and low expressiveness in terms of the provided model structures. To close this gap, this paper proposes to adapt a holistic approach for modeling components. Firstly, we investigate different dimensions of component aspects and their interrelations and secondly, we develop BA-GSem, a graph-based semantic component model for building automation. Using a case study of a multi-variant room control unit, we illustrate that the correctness of system design results can be determined more precisely when using the detailed semantic model BA-GSem. The results offer a suitable foundation for improving the quality of automated design approaches for building automation, thus facilitating the creation of modern and sustainable buildings.

1. Introduction

Building automation systems (BAS) facilitate demand-based building operation and are therefore a key concept for reducing both energy consumption and carbon footprint of modern buildings [1–3]. The application areas of smart building technologies are not limited to building automation itself, but they are also important drivers for shaping the digitalization of home health care and customized precision health care by means of smart home and assistance technologies [4]. However, the engineering of smart building systems is highly complex due to a large number of components and component variants to choose from, highly interconnected information flows and interoperability issues among components of the building system [5]. Human engineers are unable to keep up with the pace of component development in the smart building and smart home sectors: they are only familiar with a small number of the available components, which leads to sub-optimal solutions and wasted potential of energy savings [6].

Consequently, automating the engineering tasks is inevitable to cope with the growing complexity of BAS. The major engineering task is the design of the BAS, which, besides for the system development, can also be used in redesign or self-adaption [7] use cases. This component-based design process requires formal component models of high quality and expressiveness, fulfilling the following requirements: They need to offer structures with a high level of detail (*Precise Modeling*) and in order to be usable by engineering algorithms, the effort for creating and using the component models needs to be in an acceptable magnitude (*Ease of Specification and Use*). Furthermore, design algorithms will need to cope with heterogeneous levels of detail in the component database (*Robustness of Use*).

However, state-of-the-art component models suffer from low model expressiveness, focus on isolated aspects or high effort for specification and therefore fail to provide a foundation for successful automated design. This paper addresses these gaps with the following contributions:

1. **Precise Modeling:** Proposal of the *graph-based semantic model BA-GSem* for detailed modeling of component semantics in context of a holistic perspective.
2. **Ease of Specification and Use:** Identification of *important component aspects and their interrelations* as a basic prerequisite towards adopting modularization approaches.
3. **Robustness of Use:** Discussion of *the impact of different levels of detail on specific system design tasks* as foundation to assess feasibility and quality of system design.

The remainder of the paper is structured as follows: Section 2 discusses related work on formal component models for automated engineering approaches. Afterwards, the holistic perspective on component models for building automation (BA) is discussed in Section 3. Subsequently, a case study of a multi-variant generic room control unit is used in Section 4 to validate the applicability and benefits of the holistic perspective and the semantic model *BA-GSem*. Section 5 concludes the paper with a summary and an outlook towards further research.

2. Related Work

The overall task of automated design consists of fulfilling the user requirements by selecting and combining available components. Semantic specification approaches for functionality allow transforming user requirements into an intermediate, neutral and machine-understandable formal representation of required system functionality. Sub-tasks for the design of BAS are [8]: Selection of *suitable components* for system functions ("T1-ComponentSelect"), identification of required *data flow* between components ("T2-DataFlowIdent"), *interoperability*-check of all interconnected components ("T3-CheckInterop"), and ensuring that all components are capable of *providing their functionality* ("T4-CheckOperational"). Following this concept, a multi-stage *automated design approach* for room automation (RA) systems has been proposed [8, 9]. This approach is based on the common functional vocabulary of RA systems standardized by the VDI 3813-2 [10]: The first design step maps the user requirements to a formal semantic model for system functionality. In the second step, this neutral requirement model is transformed into several design suggestions using a component repository that contains formal models of the available RA components.

Existing approaches for formal device modeling include classification approaches [11–13]. These focus on isolated aspects, e. g. eProcurement or eCatalogs. Functional semantics are only annotated on a very coarse level. Several communication technologies for BA provide electronic self-description documents [14, 15]. Such documents contain models of software interfaces as well as basic functional aspects, but are technology-specific and lack detailed semantic information. Finally, Dibowski and colleagues proposed a technology-neutral semantic component model [16], specifying the device interface coarsely on both software and semantic level. Yet, the link between both levels is missing and device models are specified in a non-modularized fashion, creating a high modeling effort for different variants of a product family.

With regard to automated design, especially step *T4* is currently not sufficiently supported by any of the existing component model approaches. A holistic approach for modeling devices is required, taking into account aspects such as hardware, software, functionality, and their interrelations. We will extend the approach of Dibowski [16] to meet the practical needs of automated design approaches.

3. A Holistic Perspective on Component Models for Building Automation

3.1. Important model aspects and their relations

A general model of an automation device is shown in Figure 1. It is focused on the *Device* as the central entity and describes its inner structure as well as its interfaces. Depending on the role of the device, its

Hardware-interfaces may include connections to the process (i. e. the physics of rooms in a building), human-machine-interface elements (operating elements and display functionality), or the communication medium (automation network). The network interface is displayed on the lower right by a connection to a field bus and the appropriate *Network Stack*, which handles wrapping and un-wrapping payload with the bus' telegram structure. On the other hand, *Periphery Drivers* handle the pre-processing and integration of electrical signals received from or sent to the electric terminals (pins) of the device.

The second area of a Device's inner structure is its *Software*, which is modeled in terms of a whole *Device Application*. To allow for better modularization, the general device model allows the Device Application to be composed of individual *Software Modules* that can be managed and used independently of each other. The Device Application may restrict the number of available instances of each Software Module to account for a limited computational or storage capacity. Software Modules can also be modeled according to their interfaces and inner structure. The Software may receive and send information as payload of the communication platform (i. e. field bus). This data is shown as data points of the *Network Interface* (`nwIn` and `nwOut`). In addition, a Software Module may exchange data with the physical process by means of electrical signals, constituting the *Peripheral Interface* with the data points `pIn` and `pOut`. Lastly, a formal *Semantic Model* of a Software Module's functionality processes and generates this information on an abstract level.

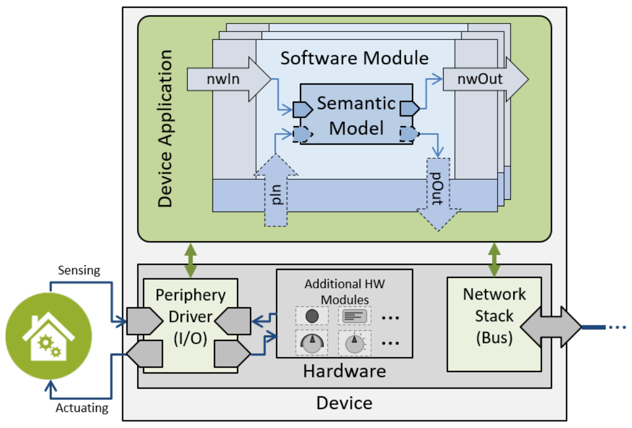

Figure 1. General Device Model of an automation device

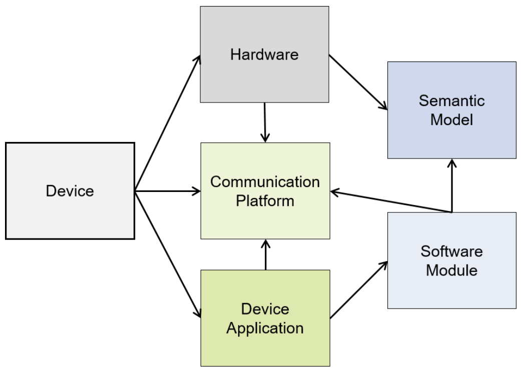

Figure 2. Model aspects and interdependencies

As can be seen from the general device model, there are several key aspects that need to be modeled for an automation device. Decomposing a device model into different aspects proves valuable if product families and therefore a great degree of similarity are involved. Instead of having a monolithic device model, each device aspect can be defined in a modular and reusable way, which makes the specification and use of device models more efficient. As part of our investigation, we examined which model parts would prove suitable for modularization. The identified aspects and their dependencies are depicted in Figure 2. A directed edge in this graph means that model artifacts of the aspect at the origin make use of model artifacts of the aspect at the target end of this edge. Starting from the *Device* as the main entity, several other model aspects are linked: Firstly, a device consists of a specific *Hardware* and a software *Device Application*. Secondly, individual variants of a product may target different communication technologies; thus, another important aspect of a device is its *Communication Platform* (which only needs to be modeled once). The communication platform is again referenced by both hardware and device application, since the first needs to contain a compatible transceiver required for communicating with this particular network and the latter makes use of the communication platform's application layer data types.

Further references can be identified from the general device model: the device application can be

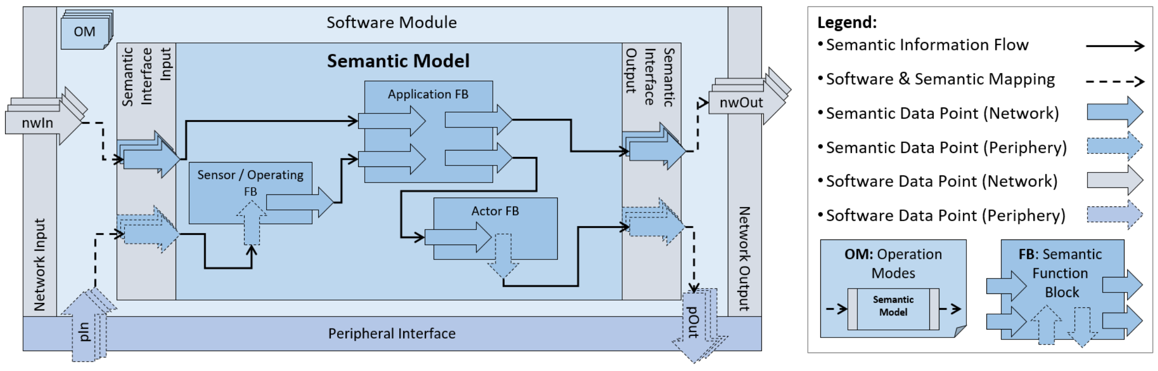
Software Module
OM
Semantic Model
Semantic Interface Input
Semantic Interface Output
Application FB
Sensor / Operating FB
Actor FB
nwIn
nwOut
pIn
pOut
Network Input
Network Output
Peripheral Interface
Legend:
• Semantic Information Flow
• Software & Semantic Mapping
• Semantic Data Point (Network)
• Semantic Data Point (Periphery)
• Software Data Point (Network)
• Software Data Point (Periphery)
OM: Operation Modes
Semantic Model
FB: Semantic Function Block

Figure 3. *BA-GSem*: Graph-based semantic model of a software module

made up out of *Software Modules*, which each contain *Semantic Models* and reference types from the communication platform. Finally, the hardware aspect referencing the semantic models is less obvious: It can be inferred from the general device model that a semantic model may only receive information from the network interface (i. e. software aspect) or from the peripheral interface (i. e. hardware aspect). Thus, a specific hardware equipment may determine, which parts of the semantic model can be chosen.

3.2. Conceptualization of BA-GSem – a Graph-based Semantic Model for Building Automation

The related literature analysis in Section 2 shows that the available structures for specifying component functionality are limited to a plain list of functions without taking into account the inner flow of semantic information as well as neglecting the distribution of data from the semantic interface to the semantic functions and vice versa. In order to provide a more fine-grained structure to model device semantics, we developed a *graph-based semantic model* for software modules of BA devices (cf. Figure 3). This model uses the function block (FB) structure proposed by the VDI 3813-2 [10]: a semantic function is represented by a specific block that defines its information flow interface. More specifically, semantic information of a certain type may be provided to the function block by directing the information flow to connectors, so-called data points, on the input side of the block. Information flows generated by the function block originate at data points on the output side of the block and can be directed to other data points. These connectors also have a specific type and the semantic linking is only valid, if the types of data points and information flow match. In case of function blocks connected to the periphery (i. e. sensor, operating, actuator, and display blocks), data points handling information flows from the peripheral interface will be drawn vertically with a dotted outline, whereas all other data points are visualized in a horizontal style with a continuous outline. This distinction is motivated by the physical connectors in [10].

With information flows connecting function blocks, a function block network as a directed graph is created. In order to be able to interact with further semantic models (contained in other devices or software modules), a semantic model also needs an interface. This semantic interface is depicted on the left (semantic input) and right (semantic output) side of the function block network. If a semantic information is created inside the function block network, this information can only be used outside the semantic model if it is represented in the semantic output interface. The semantic model is used as an abstract and formal description of how the software code of a software module processes data. Therefore, data available at the interfaces of software modules need to be mapped to the semantic interface of the semantic model, distinguishing between periphery and network. Since semantic models are more abstract than software modules, this mapping will be an abstraction (software types to abstract semantic types) or a specialization (abstract semantic types to specific software types).

Even after production and programming, automation devices can be adapted to a specific use case by configuring a set of parameters. Usually, data types can be changed and even some functionality

Table 1. Levels of detail (LoD) for model aspects

Aspect	Level	Label	Description	Example
Semantic	0	Sem0	no semantic model	-
	1	Sem1	functions annotated as *plain list*	[13], [11], [12]
	2	Sem2	additional model of *semantic interface*	[16]
	3	Sem3	detailed *graph-based semantic model*	*BA-GSem*
Software	0	SW0	no software model	-
	1	SW1	*software interface* of device application	[15]
	2	SW2	software divided into *individual SW modules*	[14]
	3	SW3	*operation modes* of software modules	*BA-GSem*
Hardware	0	HW0	no model of hardware features	-
	1	HW1	*communication technology* and transceiver	[14], [16]
	2	HW2	*number and type of peripheral connectors*	[13], [11], [12]
	3	HW3	*semantic of peripheral connectors*	*BA-GSem*

can be (de-)activated. The graph-based semantic model accounts for such post-production flexibility by the concept of *Operation Modes* (OM), which possibly alter the semantic graph, the mapping between semantic and software data points as well as their respective types in a pre-defined way. By doing so, an operation mode acts as a wrapper for the semantic model of a software module.

3.3. The impact of the level of expressiveness on the feasibility of automated design

With the proposed graph-based semantic model offering more detailed structures to model components' functional semantics, more fine-grained analyses and consistency checks become available at the level of automated design. However, it has to be expected that the semantic models in a component repository will not always be homogeneously described. The available level of detail (LoD) of semantic component descriptions in a component repository restricts, which analyses and consistency checks can be performed by the automated design algorithms.

We will now define different levels of detail for key aspects of component models (i. e. Semantics, Software, Hardware) and subsequently discuss their impact on typical automated engineering sub-tasks from Section 2. Table 1 defines the different levels of detail for the three main aspects of device models, including a short description and examples.

The first sub-task of automated design – "T1 Component Selection" – benefits from detailed semantic models since components are selected based on neutral functional requirements using the same functional vocabulary as the semantic model. Levels *Sem1* and *Sem2* already allow a proper querying of devices based on functions. *Sem3* adds the ability to query sub-graph-structures, which can be used by sophisticated device selection algorithms to solve the cover problem. The next step, "T2 Data Flow Identification" benefits from detailed models of the software and semantic aspects. The levels *SW1* and *SW2* offer a basic understanding of the software's interface; however, since they – unlike *SW3* – do not include operation modes, they incur a model error when describing the device's flexibility. On the semantic dimension, *Sem2* enables an identification, which semantic data might be communicated on a possible software connection. However, if *Sem3* is not available, there is no way of exactly assessing, what the purpose of the semantic data is and whether it is used at all.

The step that benefits the most from detailed models is step "T3 Check Interoperability". Each data flow connection needs to be investigated for validity on different levels: semantically using information starting from level *Sem2* (*Sem3* allowing more detailed checks), syntactically on software level (feasible

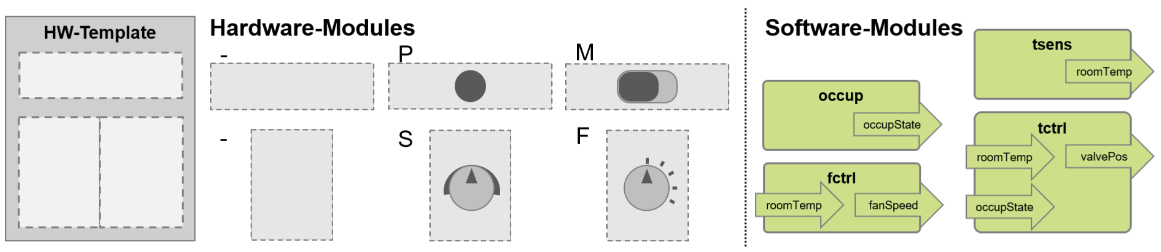

Figure 4. Building blocks of RCU Product Family – Hardware (grey) and Software (green)

at *SW1* and above with *SW3* removing uncertainty even further) and on hardware level (*HW2* and above). Finally, step “T4 Check if Operational” needs to make sure that all devices are delivered the information they require for proper functioning. This can only be assessed, if the interfaces are modeled (*Sem2* and above, *SW1* and above, *HW2* and above). The more information is available, the more accurate the operational state of components can be evaluated. For a device to be operational, it has to be ensured that all information required by semantic models is provided by either the software interface or appropriate hardware modules.

4. Case Study: Product Family "Generic Room Control Unit"

In order to provide a validation for the proposed semantic model *BA-GSem* and to show its applicability for product families in BA, we apply the method of a case study. The core element of the case study is a generic room control unit (RCU) product family, which is inspired by actual products to depict the real complexity of component models in the context of automated design approaches. The RCU product family is made up of eight similar variants that differ in their hardware configuration, which introduces subtle constraints and dependencies with respect to the functionality an RCU variant is able to fulfill.

4.1. Set-up of the Case Study

Figure 4 depicts the building blocks of the case study. All RCU variants are based on a common hardware template, which includes a temperature sensor and a small as well as a large panel for optional extension with hardware modules. There are two small and two large hardware modules: The small modules (P: *Presence button* and M: *Manual presence slider*) offer different interaction mechanisms for manually indicating the presence of persons in the room. Module P contains a button that is pressed when entering the room, while module M contains a slidable presence switch that is usually used for a distinction between day and night mode. Each large module (S: *Set point temperature* and F: *Fan speed*) features a rotary switch. Module S allows a continuous adjustment of the temperature set point (usually with an offset of ± 5K), while module F has five distinct levels for the fan speed.

Restrictions regarding the large panel are: Each module can only be used once and if the large panel is equipped, module S must be present. Furthermore, if module F is chosen, the small panel must not be empty. As can be seen, the panels cannot be considered independently from each other. In total, the RCU product family consists of eight product variants.

As can be seen from the green function blocks in Figure 4, the software for the product family has been modularized into four independent software modules: *tsens* (Temperature Sensor), *tctrl* (Temperature Control), *occup* (Occupancy), and *fctrl* (Fan Control). For simplification, the imaginary manufacturer of the RCU product family supplies each product variant with a uniform device application consisting of those four modules.

4.2. Comparison of semantic models

The comparison of different semantic modeling approaches will be presented using the example of the *tctrl* software module, since it offers a degree of complexity to illustrate issues when dealing

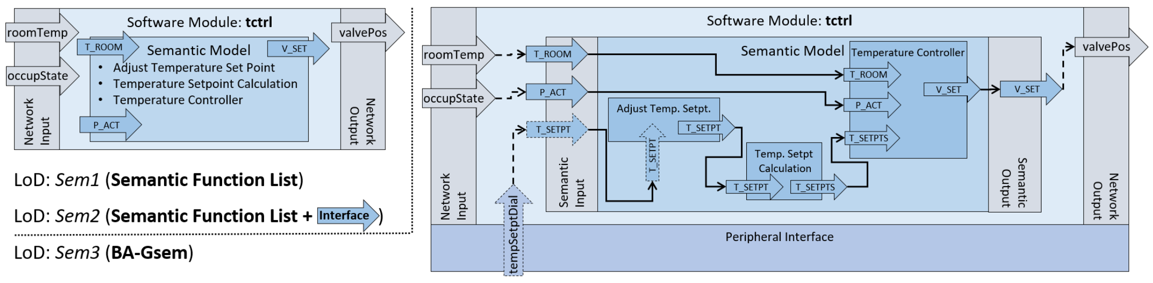

Figure 5. Comparison of Semantic Models with different level of detail (LoD)

with semantic component models. The *tctrl* software module implements a demand-based closed-loop temperature control for room temperature. The temperature controller computes a valve set point, which is transmitted from the software data point “valvePos” to a remote valve actuator using the communication network. The controller requires information about the current room temperature, the chosen temperature set point, and the current occupancy information, which is taken from both the network and the periphery: Network-wise, the room temperature and occupancy information are received by the software data points “roomTemp” and “occupState”, respectively. Finally, the temperature set point is calculated from a base set point and a set point offset that can be adjusted using the temperature set point dial of hardware module S. To convert the electrical signal from the peripheral interface, a driver block “Adjust Temperature Set Point” is required. In order to keep the example concise, a simplified notion based on VDI 3813-2 [10] is used for specifying semantics. Usually, temperature control would also involve functionality like *energy mode selection* or *function selection*, which however do not add any relevant information for this case study.

For a validation of the capabilities of *BA-GSem*, different semantic models of the RCU software module *tctrl* have been created as a manual enrichment of the existing ontology component models [16] using the OWL ontology language [17]. They are presented for each LoD from *Sem1* to *Sem3*. Figure 5 depicts these three models: As a base line, model 1 depicted on the left (without the semantic data points in blue) is a semantic model consisting of a list of semantic functions. For semantic aspects, it achieves LoD *Sem1*. The second model is an extension by adding semantic interface information (i. e. data points of semantic model). Model 2 achieves LoD *Sem2*. On the right, the third model following the *BA-GSem* approach is depicted. It consists of a graph of semantic functions and thus achieves the highest ranking *Sem3* in terms of semantic LoD. Since all variants of the RCU product family are supplied with the same device application, implicit constraints on which software modules are usable for a specific variant arise. Variants that do not have the hardware module S cannot use software module *tctrl*, since the set point offset dial is required for a proper functioning of *tctrl*. This static constraint can be evaluated at time of device modeling without need for any context information.

Furthermore, *tctrl* is dependent on a presence information being available. Thus, variants without the P or M hardware modules need to be provided with a presence information from the network. This is a dynamic constraint that needs to be taken into account during step *T4* of the automated design process – it cannot be sufficiently evaluated during device modeling.

As can be seen from Figure 5, the function-list-based semantic models on the left are not sufficient to evaluate both static and dynamic constraints. The static constraint regarding hardware module S cannot be specified at all for both LoDs *Sem1* and *Sem2*. The dynamic constraint requiring an occupancy value is also not modeled appropriately. While the software interface features a data point for an occupancy state, it is not clear for *Sem1* if this information is evaluated by the semantic model and if *tctrl* is operational without a presence information. LoD *Sem2* specifies semantic interfaces, so it is sufficient to assess that presence information “P_ACT” is required, but not if *tctrl* is operational without “P_ACT”.

In contrast, the semantic model of *BA-GSem* contains a sufficient level of detail for both constraints to be checked. In the static case, explicit modeling the peripheral interface reveals that the operational state of the semantic model depends on the availability of signals from the set point dial. For a dynamic evaluation of the occupancy constraint, it is explicitly modeled that the temperature controller is not operational without occupancy information.

In general, implicit constraints on provided functionality need to be detected and taken into account when modeling components. However, until now, functionality could not sufficiently be determined by analyzing the device software only, especially not in case of re-used uniform device applications. Using the detailed *BA-GSem* model, relevant constraints can be inferred, which allows to model modularized software components only once and without using error-prone copy-and-paste techniques. As a summary, providing additional levels of detail increases model quality and fosters the reduction of modeling errors induced by implicit constraints and inconsistencies amongst variants of product families.

5. Conclusion and Outlook

Smart building systems play a vital role for sustainable and smart cities in the future. However, the engineering of these building systems is a complex process, which still suffers from available products not being semantically modeled in sufficient detail. In this paper, we proposed *BA-GSem*, an enhanced graph-based semantic model for BA components. This model is part of a holistic perspective on component models, which also takes into account dependencies to hardware and software aspects. Thus, it is a foundation for capturing the real-world complexity of building automation product families, providing input for automated engineering algorithms. To this end, we used a realistic case study to discuss the influence of different model granularity on the steps of the typical engineering process.

Component manufacturers' expertise on product functionality and inner working mechanisms is a most valuable asset and allows for a high component model quality. The level of detail offered by *BA-GSem* implies that manufacturers need to invest more time and effort in modeling their products. While *BA-GSem* provides appropriate structures to capture this knowledge, further research should investigate, how the effort of model specification can be reduced and which quality assurance mechanisms can be put in place. To this end, we intend to provide tooling support for model specification. It is also conceivable to investigate an integration of this component model with already applied solutions for product information modeling as well as using the detailed information from the different model aspects for consistency checking. Finally, further research on how the levels of detail affect further engineering tasks is required to tap the full potential of automated engineering.

Acknowledgments
The work for this paper was funded by the European Social Fund and the Free State of Saxony (Grant no. 100310385). The authors would also like to offer special thanks to *Tuan Linh Mai* as well as *Lena Otto* and the junior research group *Care4Saxony* for their valuable feedback on the manuscript.

References
[1] Fabi V, Spigliantini G and Corgnati S P 2017 *Energy Procedia* **111** 759 – 769 ISSN 1876-6102 8th International Conference on Sustainability in Energy and Buildings, SEB-16, 11-13 September 2016, Turin, Italy URL `http://www.sciencedirect.com/science/article/pii/S1876610217302680`
[2] Royapoor M, Antony A and Roskilly T 2018 *Energy and Buildings* **158** 453 – 465 ISSN 0378-7788 URL `http://www.sciencedirect.com/science/article/pii/S0378778817318522`
[3] Stluka P, Parthasarathy G, Gabel S and Samad T 2018 *Architectures and Algorithms for Building Automation—An Industry View* (Cham: Springer International Publishing) pp 11–43 ISBN 978-3-319-68462-8 URL `https://doi.org/10.1007/978-3-319-68462-8_2`
[4] Maeder A J and Williams P A H 2017 *Studies in health technology and informatics* **245** 166–169
[5] Wollschlaeger B and Kabitzsch K 2019 *Proceedings of the 12th International Joint Conference on Biomedical Engineering Systems and Technologies - Volume 5: HEALTHINF* INSTICC (SciTePress)
[6] Welge R, Busch B H, Kabitzsch K, Laurila-Dürsch J, Heusinger S, Lipprandt M, Eichelberg M, Eichenberg E, Engelien H, Gök M, Moritz G, Hein A and Dutz T 2015 *Ambient Assisted Living: 7. AAL-Kongress 2014 Berlin, Germany, January 21-22, 2014* ed Wichert R and Klausing H (Cham: Springer International Publishing) pp 89–102 ISBN 978-3-319-11866-6 URL `https://doi.org/10.1007/978-3-319-11866-6_7`
[7] Mai T L, Lehmann M, Wollschlaeger B and Kabitzsch K 2018 *SSI2018 – 12th Smart Systems Integration Conference* ed Otto T pp 172 – 180
[8] Lehmann M, Mai T L, Wollschlaeger B and Kabitzsch K 2016 *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society* pp 7095–7100
[9] Dibowski H, Ploennigs J and Kabitzsch K 2010 *IEEE Transactions on Industrial Electronics* **57** 3606–3613 URL `http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5256330`
[10] VDI 3813-2 2011 Building automation and control systems (BACS). room control functions (RA functions)
[11] ETIM Deutschland eV 2011 Leitfaden für lieferanten zur produktdatenbereitstellung nach bmecat version 2005 Tech. rep. ETIM Deutschland e.V. URL `http://www.etim.de/fileadmin/Downloads/Dokumentation/Leitfaden_zur_Produktdatenbereitstellung_Version_2-1_2011-12-17.pdf`
[12] proficl@ss international eV 2009 Leitfaden für lieferanten zur produktdatenbereitstellung nach bmecat version 1.2 Tech. rep. proficl@ss international e.V. URL `http://www.proficlass.de/upload/pdf/proficlass-BMEcat_Lieferantenleitfaden_2_0-neu.pdf`
[13] eCl@ss center of research and development (CRD) 2014 ecl@ss-bmecat-guideline proposal for an embedding of ecl@ss into bmecat Tech. rep. eCl@ss center of research and development URL `https://www.eclass.eu/static/documents/wiki/eCl@ss-BMEcat-Guideline-2005_1_v2_1.pdf`
[14] LonMark International 2009 Lonmark device interface file reference guide Tech. rep. LonMark International URL `https://www.lonmark.org/technical_resources/guidelines/docs/LmXif4402.pdf`
[15] KNX Association 2012 Knx system specifications - datapoint types Tech. rep. KNX Association URL `http://www.knx.org/fileadmin/downloads/03%20-%20KNX%20Standard/KNX%20Standard%20Public%20Documents/03_07_02%20Datapoint%20Types%20v1.07.00%20AS.zip`
[16] Dibowski H and Kabitzsch K 2011 *EURASIP Journal on Embedded Systems* **2011** 3:1–3:17 ISSN 1687-3955
[17] Motik B, Patel-Schneider P F and Parsia B 2012 Owl 2 web ontology language. structural specification and functional-style syntax (second edition). W3C Recommendation URL `https://www.w3.org/TR/owl2-syntax/`