FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Semantic Evolution in Automation Systems Integration

DISSERTATION

zur Erlangung des akademischen Grades

Doktor der Technischen Wissenschaften

eingereicht von

Dipl.-Ing. Andreas Fernbach
Matrikelnummer 0325790

an der Fakultät für Informatik
der Technischen Universität Wien
Betreuung: Ao. Univ.-Prof. Wolfgang Kastner

Diese Dissertation haben begutachtet:

Ardeshir Mahdavi	Martin Wollschlaeger

Wien, 3. Mai 2018

Andreas Fernbach

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Semantic Evolution in Building Automation Systems Integration

DISSERTATION

submitted in partial fulfillment of the requirements for the degree of

Doktor der Technischen Wissenschaften

by

Dipl.-Ing. Andreas Fernbach
Registration Number 0325790

to the Faculty of Informatics

at the TU Wien

Advisor: Ao. Univ.-Prof. Wolfgang Kastner

The dissertation has been reviewed by:

Ardeshir Mahdavi	Martin Wollschlaeger

Vienna, 3rd May, 2018

Andreas Fernbach

Erklärung zur Verfassung der Arbeit

Dipl.-Ing. Andreas Fernbach
Treitlstraße 1-3
1040 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 3. Mai 2018

Andreas Fernbach

Danksagung

Für das Gelingen einer Dissertation sind nicht nur das fachliche Umfeld und die Betreuung von Bedeutung, sondern auch der Rückhalt im persönlichen Bereich. Das fachliche Umfeld wurde mir durch die Kollegenschaft am Arbeitsbereich Automatisierungssysteme der TU Wien in einer Form geboten, wie es geeigneter nicht hätte sein können. Freies und kreatives Arbeiten war hier dankenswerter Weise uneingeschränkt möglich. Prof. Wolfgang Kastner hat durch die gezielte Auswahl der Forschungsprojekte, an denen ich mitwirken durfte, den Weg bereitet, der schließlich durch diese Arbeit beschritten wurde. Seine umsichtige Art hat es mir erlaubt, zu jeder Zeit auf seine Expertise zurückzugreifen und dadurch immer wieder den richtigen Denkanstoß zu finden. Auch wenn die persönlichen Zusammenkünfte nur sporadisch stattfanden, hat Prof. Martin Wollschlaeger durch seine fachlichen Ratschläge dazu beigetragen, die Qualität dieser Arbeit zu erhöhen. Ich bin froh und dankbar, im Zuge dieses Lebensabschnitts auf die genannten und die vielen hier ungenannte Kollegen getroffen zu sein.

Die Momente im Privaten zu tolerieren, die durch meine Gedanken an diese Arbeit geprägt waren, war das Los meiner Lebensgefährtin Claudia. Danke Dir für Deine Unterstützung und Deine unerschöpfliche Geduld!

Kurzfassung

In modernen Gebäuden sorgt eine komplexe Haustechnik für die Bereitstellung angemessener Bedingungen. Zu den hierzu wesentlichsten Gewerken zählen Beleuchtung und Verschattung sowie Heizungs-, Lüftungs- und Klimatechnik (HLK). Für einen effizienten Betrieb dieser Anlagen sorgt eine Vielzahl an Gebäudeautomationstechnologien und Systemen, die ihrerseits auf bestimmte Gewerke optimiert sind. Zusammen ergibt sich eine durch starke Heterogenität geprägte Systemlandschaft. Um dennoch Interoperabilität zwischen diesen zueinander inkompatiblen Technologien zu erreichen, wurden in der Vergangenheit verschiedene Lösungsansätze zur Integration entwickelt. Hiervon zeichnete sich die Einführung einer übergeordneten Abstraktionsschicht, die einen vereinheitlichten Zugriff auf die verschiedenen Gebäudeautomationssysteme ermöglicht, als der geeignetste Ansatz ab.

Diese Dissertation greift die beschriebene Situation auf und beschreibt ein Verfahren zur Integration von Gebäudeautomationstechnologien, das gleichzeitig auch Informationen über das Gebäude selbst berücksichtigt. Dazu wird ein Ansatz vorgestellt, der zwei Integrationsebenen unterschiedlicher Komplexität und Ausdruckstärke umfasst. Um die für diese abstrakten Darstellungen geeigneten Methoden zu finden, werden zunächst aktuelle Sprachen zur Wissensdarstellung hinsichtlich ihrer Eignung untersucht. Informationsmodelle für ausgewählte Technologien werden mit Hilfe des Standards OPC Unified Architecture (OPC UA) erstellt. An dieser Stelle wird auch eine Schnittstelle zur Einbindung von Systemen der Industrieautomation geschaffen. Hieraus ergibt sich die erste Integrationsebene.

Diese OPC UA-basierten, technologiespezifischen Informationsmodelle werden in einem nächsten Schritt in einen umfassenden Gesamtkontext gesetzt, der Aspekte der Bauwerksdatenmodellierung miteinbezieht. Diese holistische Sicht wird durch eine Ontologie, die in der Web Ontology Language (OWL) verfasst ist, dargestellt. Die daraus resultierende Wissensbasis repräsentiert nicht nur statisches Wissen über Gebäude und deren Automationssysteme, sondern erlaubt auch Zugriff auf Laufzeitwerte der entsprechenden Datenpunke. Damit ist die zweite Integrationsebene definiert. Für den Zugriff auf die Wissensbasis wird eine semantische Abfragesprache verwendet.

Der Entwurf von Gebäudeautomationssystemen und die hierbei zu berücksichtigenden Informationssicherheits-Maßnahmen bilden den Rahmen um die vorgestellten Integrationsansätze. Es wird ein durchgängiges, teilautomatisiertes Verfahren für den Entwurf

dieser Automationssysteme und der zwei darüber liegenden Integrationsebenen vorgestellt. Die Anwendung dieses Verfahrens wird anhand repräsentativer Vertreter der jeweiligen Ebene gezeigt. Da Informationssicherheit in integrierten Gebäudeautomationssystemen eine essentielle Rolle spielt, werden dahingehend für die verwendeten Technologien OPC UA und Web services (WSs) die zur Verfügung stehenden Möglichkeiten untersucht. Schließlich werden Modelle zur Herstellung von Vertrauensbeziehungen zwischen den beteiligten Komponenten evaluiert.

Abstract

Modern buildings are conditioned by a variety of building services. These mainly involve lighting and shading as well as Heating, Ventilation and Air Conditioning (HVAC). To assure an efficient operation of the services, the field of Building Automation Systems (BASs) has established. These systems are nowadays characterised as a highly heterogeneous landscape of different standards and technologies. Each of these technologies in use has its dedicated fields of applications in controlling building services. The resulting heterogeneity leads to considerable interoperability issues between building automation technologies. To address these issues, several approaches of integration have been developed in the past. The most promising way which has emerged was to provide a unified abstraction layer on top of these mutually incompatible systems.

This thesis captures the current situation and presents an integration method for BASs but also including information about the building itself. Therefore, an architecture of two levels of comprehensiveness is chosen. To find the proper means to express the desired abstractions, state-of-the-art knowledge representation languages are examined with respect to their suitability towards creating unified views on the mentioned heterogeneous information sources. Information models are created for selected representatives of building automation technologies by using OPC Unified Architecture (OPC UA). Additionally, an interface to integrate technologies from industry automation is defined at this point. This constitutes the first level of integration.

The OPC UA-based, technology-specific views are in the following raised to a bigger context including aspects of Building Information Modelling (BIM). The resulting, holistic view is established by a Web Ontology Language (OWL) ontology which does not only regard static knowledge about buildings and their embedded automation systems but also allows access to runtime values of datapoints in the BASs by a semantic query interface. Hereby, the second level of integration is established.

As a frame around the developed integration methods, automation systems engineering and security engineering are addressed. A workflow of semi-automated engineering throughout the hierarchy of automation systems and the two integration levels is proposed. The application of this workflow is shown by means of representative technologies from every layer. Since security plays an essential role in integrated BASs, OPC UA and Web service (WS) technologies are examined with regard to this aspect. Applicable models of establishing trust in these environments are finally evaluated.

Contents

CHAPTER 1

Introduction

1.1 Motivation

Building Automation Systems (BASs) have been continuously developed starting from the end of the 19th century. Within this history, a very heterogeneous landscape of standards, technologies and proprietary solutions has arisen. The solutions deployed are each specialised on a distinct field of application (i.e., trade) in a building which makes it often unavoidable to deploy more than one technology within an installation of a single building. These subsystems cannot be considered functionally isolated as it will be illustrated later by typical use cases. In order to allow interaction within these heterogeneous systems, efforts must be undertaken to create interfaces allowing information exchange across the borders of the subsystems. There exists a number of possible solutions to achieve this so-called integration, addressed later in this thesis.

1.1.1 The historical development of building automation

The history of building automation and control systems began in the late 19th century. Back then, it was common practice that the caretaker walked through the building from time to time checking and noting the temperature in each room. When the first automated heating control systems appeared on the market, the goal was to take over the so-far manual operation of boilers and heat distribution. Increased comfort was the main argument for the first BAS devices. According to contemporaneous product advertisements, [1] these systems could maintain room temperature with an accuracy of two degrees off the setpoint. The Johnson Controls "Automatic Temperature Control System" is taken here as an example. Its main components were room thermostats and actuators for valves and dampers interacting pneumatically. The valve actuators adjusted the heat water or steam flow to the radiators, whereas the damper actuators set the inlet of warm air to the rooms or the amount of combustion air to the furnace. Figure 1.1 shows an illustration of some Johnson components and a schema of their possible

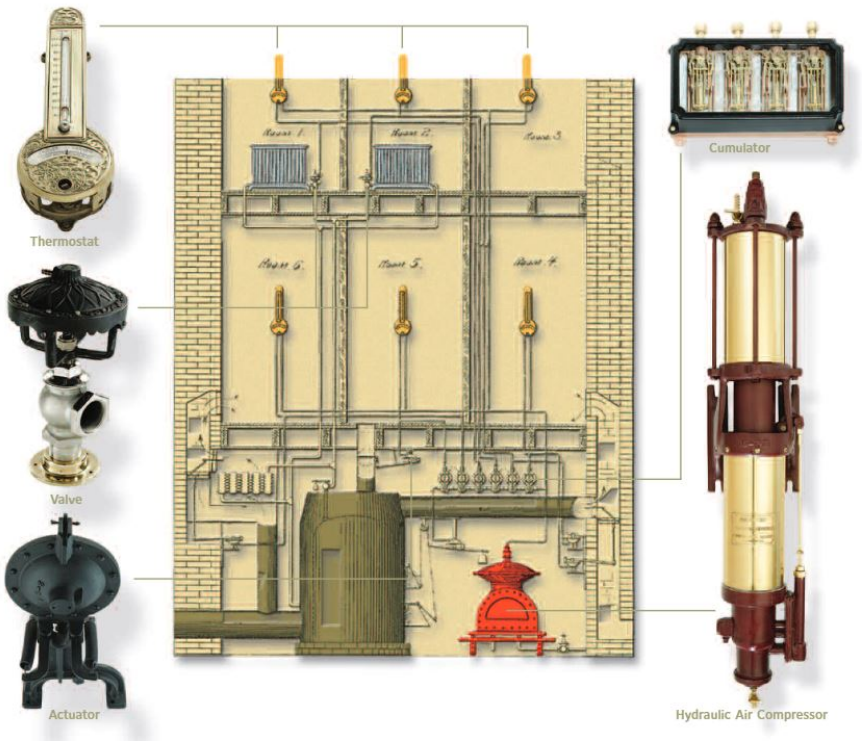

Figure 1.1: Johnson system of temperature regulation [1]

application in building. The "Cumulator" in the right upper corner of the figure adds up the heat demand from the thermostat and sets the combustion air damper at the furnace accordingly. The compressed air necessary for the system to operate was provided by a compressor working with the power of mains water.

Since the complexity of these early temperature control systems more and more increased, the necessity arose to monitor and to manipulate all of a building's control devices from a single point. A single person could now overview the conditions in a facility and the state of the installed systems. These pneumatic control centres from the 1960s were soon replaced by ones with mixed pneumatic and electric operation. The actuators were driven pneumatically by transducers where the feedback signalling from sensors and the controller were replaced by electronic - but still analogue - components [2]. Digital technology entered this domain in the 1970s, where mini-computers were used to control building systems. Programmable, microcontroller-based devices, which were already in use in the industry for process control (i.e., Programmable Logic Controllers (PLCs)), found their way to the building control sector in the 1980s. This concept is still known today under the name Direct Digital Control (DDC). These DDCs carry out dedicated automation tasks like PID controllers or sequential control of their subsystems. Their domain was the control of the building services Heating, Ventilation and Air Conditioning (HVAC) - the major application in building automation for a long period. At the control

centers, also minicomputer-based solutions were introduced. They facilitated a central management of events and failures, trend recording and evaluation with the goal to optimise system operation. Initially motivated by the oil crisis in the 1970s, where energy prices significantly climbed, proper use of energy in building operation became a topic. This circumstance led to the development of Energy Management Systems (EMSs). These software-based solutions had the main goal to preserve fuel by means of optimal and demand-oriented system operation. The management of setpoints of the underlying DDCs by following occupancy schedules or the optimised start and stop of plants depending on indoor and outdoor conditions are still classic EMS tasks.

Soon, networked connections were established among DDCs and management stations as a result of the new demands in communication relations. Especially the distributed system architecture which evolved from centralised approaches requires efficient data communication to bridge the distances between the spatially separated devices. Various protocols which were already established in the business world found their application in the HVAC sector. Attached Resource Computer Network (ARCNET), Ethernet and RS-485 were the technologies which formed the first LANs dedicated to this field. With the progress of IT technology, also building automation components became more powerful and intelligent, both with regard to communication and processing power. Demand on new components like on field devices or Human Machine Interfaces (HMIs) was satisfied by industry automation parts. Modern controller devices are usually equipped with IP connectivity which enables the use of standard IT infrastructure components like Ethernet switches and wiring. This way, the integration of building automation networks into the office and PC-based world is facilitated.

Modern BASs are heterogeneous networks composed of a multitude of different technologies and standards. In contrast to their historic predecessors, they fulfil a broad variety of tasks in a building. Providing proper inside conditions by means of HVAC systems is still a major requirement, but high effort is also put into providing visual comfort with the help of lighting and shading applications. Today's buildings are usually equipped with safety-relevant applications like emergency lighting, fire alarm systems, extinguishing systems and contaminant detection systems (e.g., for gas or carbon monoxide). Physical access control in combination with intrusion detection as well as surveillance systems are also relevant for functional buildings.

Each of the applications named above can be assigned to distinct areas of the three level functional hierarchy of BAS. This model is defined in the DIN EN ISO 16484-2 [3]. An illustration thereof can be found in Figure 1.2. At the lowest level of this pyramid-shaped model, the *field level* sensors collect data and actuators interact with the physical process when sensing, metering, switching and setting takes place. At the *automation level* located in the middle of this three level hierarchy, data from the lower tier are processed, control loops are executed and the resulting values are fed back to actuators. At the top level, the *management level* of BAS, setpoints are defined, centrally accessible process data are visualised and archived for trending and monitoring applications. At this point, data originating from different automation network standards located at the lower two

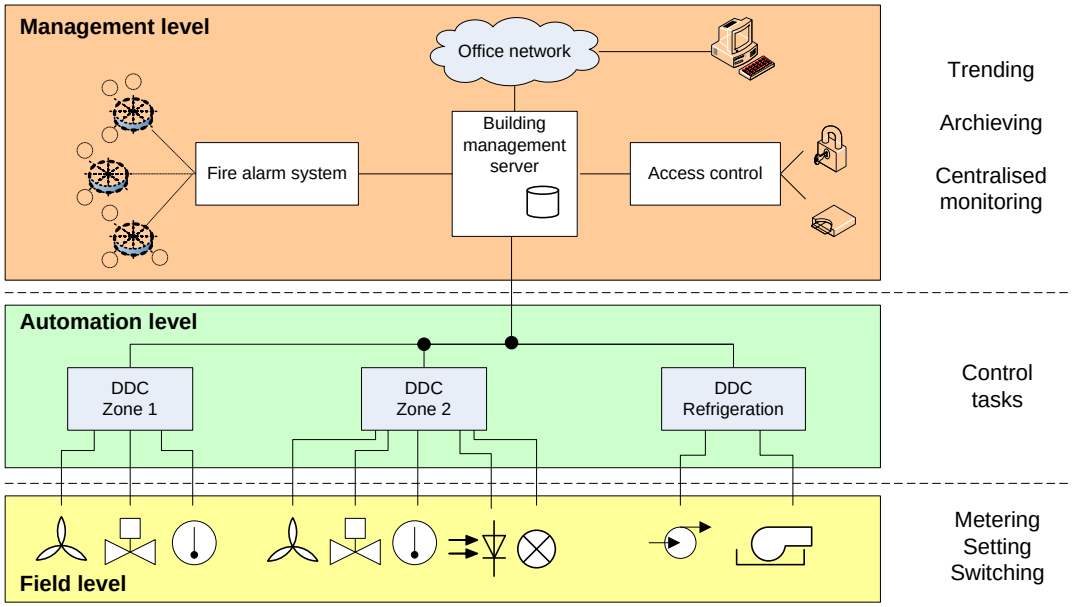

Figure 1.2: Three level model of BASs, adapted from [4]

levels of a BAS are aggregated.

These top-level applications are usually subsumed by the term Building Management Systems (BMSs). Sometimes, the term Building Energy Management Systems (BEMSs) is used with a similar meaning [5]. This apparently reflects the fact that assuring energy efficiency [6] is one of the major current goal of these applications. BMS/BEMSs can be compared to Supervisory Control and Data Acquisition (SCADA) applications known from factory automation due to a high overlap of functionalities. Central access is provided to the operator via a Graphical User Interface (GUI) to most of the subsystems' datapoints. Live values can be monitored and are also recorded for archiving. These historical values can be used for accounting of energy consumption, long term error detection and optimisation. Besides the manual adjustment of setpoints, time programs (i.e., schedules) can be defined for repeating occupancy cycles of rooms and building areas. A typical scheduler influences setpoints for temperature and air exchange rate in accordance with office or work hours. In case of an error in one of the underlying systems, the operator needs to be informed in order to take proper actions. This is achieved by alarms (i.e., critical events) generated by the subsystems which are propagated to the BMS and indicated to the operator.

One relatively new application for BMSs is the integration of meter points with the help of smart meters capable of remote readout. Usual meter points in buildings and factory plants accumulate the consumption of water, gas, electricity, compressed air or heat. A segmentation of metering the consumption according to floor, building parts or plant segments is usual practice. Today's smart meters provide remote readout interfaces

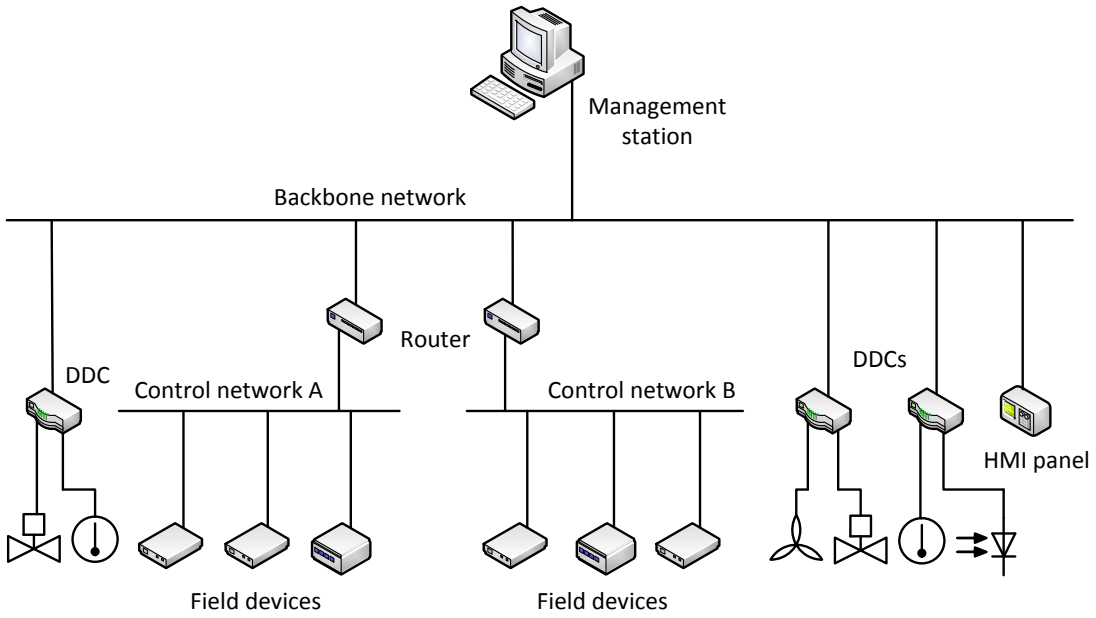

Figure 1.3: Two-tier network architecture [12]

using various wired and wireless readout protocols like Meter Bus (M-Bus) [7] or the ZigBee-Smart-Energy-Profile [8]. A BMS collecting the metering data is able to perform analysis of consumption values like a comparison on weekly, monthly or seasonal basis.

Regarding the hierarchical depth of BASs, practice has shown that implementing a 3-tier functional architecture (according to ISO/DIS 16484-2 [3]) is appropriate only in rare cases. Instead, the functionalities modern control devices are capable to result in an architecture with two levels of networks. Figure 1.3 shows an example of such an architecture. Typical field devices with control network connectivity (e.g., KNX [9] or LonWorks [10]) like switching actuators, room temperature sensors or control panels already implement automation level functionalities to be able to perform dedicated control tasks like temperature regulation or lighting scene control. These low-bandwidth control networks are sufficient for the typically small amounts of data (sensor readings or control commands) to be exchanged between field devices. Cost-efficiency with regard to the communication medium, easy wiring combined with supply lines and wire-saving bus topologies also play a role. Often, one control network segment is limited to a building floor or zone. These networks are interconnected via routers to a backbone network of higher bandwidth. Here, IP over Ethernet is the usual network technology. Central control, monitoring and diagnostics is hereby enabled. For nowadays DDCs, where the Building Automation and Control Network (BACnet) [11] is a well-established technology, an IP interface is standard. This allows to directly connect DDCs into the backbone network while field devices are directly coupled to a DDC typically using standard signals like 24V, Pt1000 or 0-10V.

The challenge is to find a common software interface to these different network standards which follow a different way of representing and transporting process data. The classical, but not very convenient way to reach this goal is to integrate technology-specific interfaces (i.e. drivers communicating with the underlying networks) in the management level applications. The disadvantages of this approach are a lock-in to distinct technologies and little flexibility for future extensions including new network standards.

Especially at the management level but at the lower levels of BAS as well, IP-technologies are already in use. KNXnet/IP, BACnet/IP [11] and LonWorks/IP tunnelling may be called in this context. This already existing IP infrastructure facilitates the use of Web services (WSs) [13] as an approach to achieve the desired interoperability. Since WSs are independent of any hardware platform or operating system, they provide a convenient way to access runtime data in a common and unified way.

1.1.2 Integration approaches

Automation systems both in the industrial and the building domain follow a hierarchically organised architecture. This circumstance was already recognised in the 1970s where the first functional models were introduced with the aim to structure the increasingly complex systems in production plants. A property of these early models was the strict separation of single layers of a hierarchy with regard to functions and communication networks. The result was a nowadays very common model, the automation pyramid. The hierarchical order of the layers is also reflected by a number of characteristic properties like the number and complexity of devices at a distinct layer, data throughput and timing constraints. Each level of the automation pyramid has its dedicated functions and networking. At the bottom margin of the pyramid the physical environment (i.e., the technical process) is situated whose properties are captured and influenced by the devices of the field level. These sensors and actuators have connectivity by means of analog standard signals (direct wiring) as well as narrow-band fieldbus networks. Hereby, the data exchange between sensor/actuator systems and controllers located at the following levels is enabled. At these process and cell levels, autonomous control tasks are executed in controllers commonly known as PLCs, automation stations and industrial PCs for Manufacturing Execution Systems (MESs). Their main tasks are scheduling and dispatching of jobs. Ethernet-based communication is now dominating these levels in form of various technologies which are partly also capable of real time. They are subsumed by the term industrial Ethernet. Standard IP networking is established at all the upper layers which cover tasks concerning shop floors, factories and cross-company applications. The according applications are SCADA and Enterprise Resource Planning (ERP) up to supply chain management. The functions which need to be realised at these levels are basically scheduling, planning and monitoring - including data archiving. Communications takes place via office IP infrastructure with a scale from locally limited networks (LANs) up to Internet communication (WANs).

This original automation hierarchy models defined as a stack of four to six different functional levels is not applicable to modern plants anymore. The wide-spread establish-

ment of Internet technologies resulted in a collapse to a two or three level architecture consisting of a planning level on top, an intermediate one where control tasks are carried out and the field level.

In order to allow information exchange across the borders of network and automation technologies, translations (either unidirectional or bidirectional) between the respective world models of protocols need to be defined. The basic semantics of the protocols used in automation systems is rather similar: the exchange of datapoint values and issuing control commands. However, different communication patterns are used like, e.g., client - server, publish - subscribe or producer - consumer. A classic gateway device connects two different network protocols at the application layer [14]. This - historically seen - early solution leads to a number of disadvantages, especially in highly heterogeneous systems with a high diversity in technologies. In the industry, this is called the "Inter-enterprise Nightmare" [15]. Figure 1.4 shows the situation where point-to-point integration is carried out. The subsystems illustrated on the lower level each use their own technology which needs to be integrated into the high-level applications on top. In the worst case, pair-wise integration with $\binom{n}{2}$ mappings is carried out. [16] This leads to multiple dependencies resulting in poor maintainability and extensibility. Often, many different vendors are involved creating custom made solutions using their proprietary technologies. A vendor lock-in is often the case when architectures like these are set up.

An alternative, which appeared to overcome the disadvantages of the gateway approaches is the concept of middlewares. The goal behind these developments was to provide a homogenisation between communication and application as well as among applications. First candidates were Common Object Request Broker Architecture (CORBA) [17] and Microsoft Distributed Component Object Model (DCOM). Today, Web service-based middleware dominates the field which is justified by this paradigm's platform-independency, flexibility and network-friendliness with regard to firewall configurations. The state-of-the-art in integration by WSs will be discussed in Section 2.2 in more detail. The principle of homogenisation by creating a unified model is shown in Figure 1.5. The unified information model on top reflects the information representation of a middleware solution. An information model is a pattern consisting of objects and relations which can be instantiated by information originating in a concrete reality. It shall provide adequate expressiveness depending on the technologies to be integrated and the intended environment of application. Therefore, it needs to cover distinct requirements on semantics which can range from only a simple datapoint model to a comprehensive form of knowledge representation. The information models at the bottom of the illustration reflect the means how information is represented by automation technologies to be integrated. Technologies in use in the field of automation follow individual approaches which lead to individual mappings between their information models and the middleware information model. Therefore, model transformations ($\mathrm{TR}_1 \ldots \mathrm{TR}_n$) need to be defined which are usually a set of rules describing how entities from a distinct technology are transferred to the unified information model. Depending on the formats involved in this process, the rules can be written in a machine-readable form. Once defined, the rules can

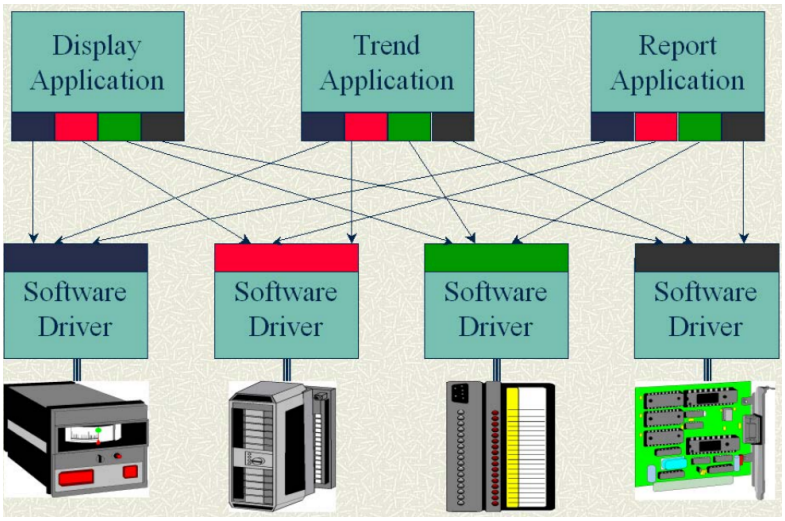

Figure 1.4: Inter-enterprise nightmare [15]

be executed automatically on arbitrary instantiations of both models. Machine-friendly formats are, for example, XML-based description languages.

Integration can be seen in two dimensions with regard to the automation pyramid - horizontal and vertical. Horizontal integration encompasses the interconnections at one level which inherently leads to distinctions between the conditions at the different levels. Contrary to upper levels where exclusively IP-based communication takes place, especially at the field level a variety of different field bus systems can be found which differ in their protocol stacks. This circumstance makes a large difference when an integration solution like a middleware is designed. Vertical integration takes place across different levels of the automation hierarchy. One major aspect is the interconnection of fieldbus networks and IP-based networks. At the upper tiers, mainly the application layers need to be concerned since network environments are fully IP-based. Therefore, interoperability up to the network layer is already assured at these levels of the pyramid model. Here again, middleware approaches come into play since they are a powerful and flexible method for vertical integration. They therefore play an essential role within this work. The progress of automation technology development and integration approaches up to the newest trends can be traced by the following literature: [18], [19], [20].

As mentioned before, when introducing a middleware integration approach, the central question is the way how information is represented at this newly established homogenisation layer. The unified model which is instantiated can be of arbitrary expressiveness, depending on the requirements to the representation. In the following, these requirements will be derived from the intended use cases to be realised on the basis of an integration layer.

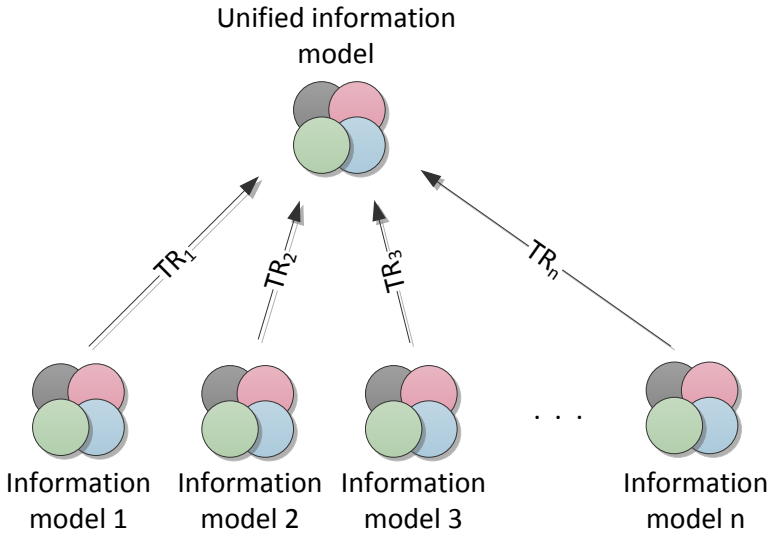

Figure 1.5: Information transformation

1.2 Problem statement

Automation technologies have not only evolved in the building sector in the past decades but also (and in a much more extensive and faster way) in other domains. The industrial manufacturing domain is not imaginable without automation, Computer Aided Manufacturing (CAM) methods are state-of-the-art in most plants. Currently, the technological development is further accelerated by the "Industry 4.0" doctrine by which information and communication technologies are further pushed into production systems [20]. Also the electrical energy supply sector is currently seized by the progress of automation and IT systems. This concept, where power grids are overlaid by an IT infrastructure is called smart grid [21], [22] and plays a crucial role when exploiting renewable energy sources [23]. Electrical mobility is tightly related to intelligent grids since control systems are necessary to coordinate the high loads which are put on the grids when charging vehicle batteries.

All these domains are inter-related to each other and they already have points of interactions where cross-domain data exchange takes place. It is expected that the realisation of innovative use cases in the near future will require cross-domain interaction of automation systems in a much higher degree. The main drivers for these expected future use cases are optimised utilisation of existing systems like it is the case for smart metering as well as energy savings by exploiting synergies which arise when interconnecting domains. These developments are in the meanwhile even driven by official authorities like the National Institute of Standards and Technology (NIST) with regard to smart grid and BASs interaction [24]. In the following, a number of such use cases is sketched.

In the course of the optimisations towards minimal energy demand of buildings, in many

cases interworking between different trades is required. It is normally the case that in different trades, different technologies are applied. For example, primary plants for provision and distribution of heat, cold and fresh air (HVAC) are often equipped with BACnet controllers. Lighting and shading applications as well as room controlling often base on KNX systems, whereas for the readout of meters counting electric energy, water and gas demand, M-Bus is an often-used protocol.

A simple scenario where KNX and BACnet components need to interact is the determination of the overall fresh air, heat and cooling demand in a building in order to calculate the setpoints for the HVAC primary plants. Information from presence detectors and room controllers (i.e., room control) implemented in KNX networks needs to be collected and passed to a network of BACnet controllers operating the primary equipment, such as boilers, chillers and air handling units. By processing the demand requests from the room controllers, fast reaction of the primary plant is assured. Also an intelligent shading system can make use of such an integrated system. The state of shutters, lighting, heating and cooling systems are coordinated such that least possible energy is consumed. Cooling demand for preventing summer overheating is minimised by finding the proper trade-off between shading and lighting. In order to minimise heating energy demand in winter, maximum use of solar heat gains is made.

For Energy+ buildings (i.e., buildings which deliver more energy to the grid than they consume) or - in the future - when even Energy+ city districts become reality, integration of BASs into smart grids is a logical consequence. Energy providers will changes tariffs for electricity (maybe also for gas, district heating or cooling) over daytime depending on grid load or renewable energy supply. Smart meters play a central role in these macro systems. They are used for data acquisition on energy consumption but also the building-integrated production of renewable energy like solar heat or photovoltaics. In such innovative buildings, energy buffers for heat and electric energy are deployed which are integrated in the building's plants. The utilisation of these buffers needs to be coordinated with the current and expected energy demand and production as well as with tariff information from the energy provider. M-Bus enabled smart meters act as information source for current demand, production and tariffs which is processed by a BEMS. The latter influences e.g., a BACnet heat pump controller depending on energy price and current heat buffer level.

From these scenarios described above, the following requirement on the BAS can be derived:

- **Requirement 1**: An infrastructure shall be created which integrates BAS technologies in a way that seamless information exchange is possible independent of the automation technology.

Whenever industry appliances and buildings are located close by (which is often the case at a company site), certain synergy effects arise, e.g., a potential for energy saving. This

is especially the case if the actual production process is very energy intensive. Examples for such processes can be found in the petrol, steel and paper industry. Scenarios where synergy effects can be utilised are the use of waste heat for building conditioning and the possibility to avoid load peaks in electrical energy demand. Also optimising the use of in-house power plants supplying both the factory plant and buildings must be considered. This opens an additional technology domain to be integrated into a management system which performs such coordination tasks, namely the field of industrial automation. Like in BASs, a number of well-established technologies exists in industrial automation which have a focus on dedicated applications associated to a distinct section of the automation hierarchy.

- **Requirement 2**: The infrastructure from Requirement 1 shall be extended such that access to systems from both industrial and building automation domains is enabled in a unified way.

For management applications, the spatial and topological locations of datapoints, plants and devices are essential. In case of maintenance, troubleshooting or damage this information facilitates an efficient workflow. Service personnel can quickly be sent to the location of the defective part in order to perform the required actions. Also spatial or topological correlation of failures can be evaluated to gain an indication of their cause.

- **Requirement 3**: The integration infrastructure shall comprise spatial and topological information associated with runtime and device information.

When extending the perspective from automation systems integration to a holistic view on a building including detailed, full parametric information about the building itself and its embedded automation systems, a new level of application scenarios can be realised. In turn, the amount of information to be handled by such a system drastically increases which makes - compared to classical manual exploring or browsing - new methods of access necessary. This includes requesting information on arbitrary entities, either of static or runtime character, and performing logical conjunctions as well as arithmetical operations and filtering.

- **Requirement 4**: A way of representation shall be established with the capabilities to map a holistic view of a site including building information and the embedded automation systems. In order to cope with the expected large scale of information, this representation shall also be able to process semantic queries.

When it comes to BASs design and engineering, a high amount of human contribution is involved. Currently, this process includes manifold repetitive manual tasks due to a lack of Machine-to-Machine (M2M) interfaces between the different phases of planning and

engineering. Therefore, manual propagation of information is necessary throughout the chain of used tools. This workflow is considered as error prone where in frequent cases information is lost or misinterpreted. Additionally, this approach is inflexible to changes on early design steps at a late design phase due to the required manual re-propagation of these changes. Manual work during the design process needs therefore to be reduced to a minimum by assuring that information is only entered once by the user during the workflow and automatically reused wherever it is required.

- **Requirement 5**: A concept shall be defined which supports building automation and integration engineering by avoiding manual information transitions between the involved intermediate steps.

The today's awareness about the need of inherently secure systems leads the principle of "security by design". Automation systems engineering therefore needs to regard, besides functional aspects, also security considerations. This especially applies to the networks and components on IP level which are this way easily accessible for potential attackers. An even higher risk exists for integration platforms where aggregated information from the subsystems is present.

- **Requirement 6**: For the integration infrastructure to be designed, state-of-the-art security measures shall be deployed. This especially includes measures for establishing trust relationships between the involved components.

1.3 Methodology

Integration by middleware may take place on different levels of the automation systems hierarchy and may therefore handle various sources of information. In the base case, only information from automation technologies is regarded. Depending on the use cases to be realised, it is necessary to additionally include information about properties of the system, the plant or the building right up to environmental conditions or user behaviour. In other words, in a first step, heterogeneous building automation technologies are integrated. In the following, this matter is called *first level integration*. In a second step, this already homogenised information is put into an even larger context, which will be named *second level integration*. This two-stepped integration approach affects the requirements on the forms of knowledge representation used on each level. It needs to be determined, which methods of knowledge representation are suitable to achieve both levels of integration. Also, non-functional requirements like usability and tool support are of relevance. Therefore, the question to be answered is:

Which state-of-the art technologies are applicable and provide the desired, tailored expressiveness for each level of integration? Moreover, how can a secure operation and an efficient engineering workflow be assured?

1.3.1 Modelling workflow

The processes of defining the required knowledge representations on the different levels of integration require to follow distinct methodologies. The chosen workflow is mainly taken from [25] and from the classic software engineering process which also follows the described phases. In the *Analysis* phase, the stated requirements are examined in a first step. The result is a fine-grained catalog of functional and non-functional attributes. An essential point is also the study of preliminary and related work. The goal is to build upon existing concepts in the highest-possible degree in the following. The *Design* phase starts with defining the main modules of the knowledge representation to be created. In this chosen top-down process, the main modules are refined and filled with the required information entities. Hereby, an important aspect is to comply with existing naming conventions from the domain. A usual approach is to first define the necessary terms and then establish the relations between them. In the *Implementation* phase, the mapping of the result from the design phase to the actual form of knowledge representation takes place. Hereby, tool support is an important aspect. The final *Evaluation* phase determines to which degree the developed knowledge representations meet the defined requirements. Therefore, completeness is checked, i.e., it is revised if the required information is regarded to enable the intended scenarios. In case of incompleteness, the missing entities are complemented. In practice, an iterative process has proven most convenient here.

1.3.2 First level integration attributes

The requirements on first level integration middleware can be summed up by the following properties:

- Design of a meaningful datapoint model including modelling concepts for data encoding and engineering units
- Definition of an appropriate device model
- Development of a plant and building topology model
- Definition of a concept for runtime interaction with automation systems
- Provision of an interface to iteratively browse resources

1.3.3 Second level integration attributes

Middleware and integration solutions operating on building or plant-wide level (i.e., performing second level integration) aggregate a high amount of information from their subsystems which is characterised highly eclectic (complex systems).

- Extension of the first level integration datapoint model towards holistic representations

- Development of a concept for runtime interaction with the first level integration middleware
- Definition of an automatic transformation process for information from first level integration
- Ability to maintain a large amount of information, i.e. assure scalability
- Provision of an interface to perform semantic queries
- Capability to perform logical, algebraic and filter operations on the query results

1.3.4 Non-functional attributes

Independent from the individual requirements to the forms of information representation, a number of general properties must be regarded:

- Correctness of the representation, regarding both syntax and semantics
- Efficiency in processing and reasoning
- Complexity of used mechanisms and the shape of the learning curve resulting thereof
- Translatability to other formats to enable easy M2M interaction
- Security and access control mechanisms to assure confidential and authorised use of the system
- Community and tool support

Therefore, different methods of knowledge representation need to be examined to find the adequate formalism for each system level. The aim is also to rely on industry and IT standards which are open, well supported and with a long expected lifetime. A unified exchange format for configuration and engineering information is also required.

1.4 Structure of the work

In Chapter 1 of this thesis, the topic area of BASs is unrolled. A brief overview on the historical development which lead to the nowadays' heterogeneity of these systems is given. The challenges resulting thereof are addressed and requirements on modern installations are derived. The methodology how this thesis elaborates these requirements is described in the following.

A comprehensive state-of-the-art analysis of topics relevant for this thesis is given in Chapter 2. This includes mechanisms for knowledge representation and comparison of

their key attributes. The resulting, most-suited integration technologies are described subsequently. The following section is also dedicated to engineering support techniques. The chapter closes with an overview on related scientific work.

The following three chapters address the problems stated in Chapter 1. First level integration concepts are demonstrated in Chapter 3 on a number of prominent representatives of building automation technologies. This furthermore touches the smart grid and industrial automation domains. In Chapter 4, the focus of integration is expanded to holistic representations of buildings and their automation systems. Semantic Web technologies are the method chosen therefor. Methods for facilitating the engineering processes in the introduced multi-level infrastructure are another contribution of this thesis which is presented in the first section of Chapter 5. Addressing security requirements in BASs integration and establishing trust are the central topics of the second section of this chapter.

In the final Chapter 6, the research question stated initially is picked up again and put in contrast to the contributions of this thesis. The results are discussed and an outlook to future work is given.

1.5 Contributions

This thesis has been carried out under the TU Vienna's doctoral college of Environmental Informatics. A number of research projects provided the context for the publications written by the author. The most-related projects are "Web-based Communication in Automation" (WebCom), "Information Modeling in Automation" (iModelA) and "Secure and Semantic Web of Automation" (SeWoA), all funded by the Austrian Research Promotion Agency. In the following, a selection of the most-relevant publications of the author is given:

A. Fernbach, W. Granzer, and W. Kastner. Interoperability at the Management Level of Building Automation Systems: A Case Study for BACnet and OPC UA. *Proc. of 16th IEEE Conference on Emerging Technologies and Factory Automation (ETFA '11)*, Septempber 2011.

A. Fernbach, W. Granzer, W. Kastner, and P. Furtak. Mapping ETS4 Project Structure to OPC UA using ETS4 XML Export. In *KNX Scientific Conference*, nov 2012.

A. Fernbach and W. Kastner. Certificate Management in OPC UA Applications: An Evaluation of different Trust Models. In *Proc. of 17th IEEE Conference on Emerging Technologies and Factory Automation (ETFA '12)*, 2012.

A. Fernbach and W. Kastner. Information modelling in OPC UA. In *Tagungsband SPS/IPC/Drives*, 2012.

A. Fernbach and W. Kastner. Integration of Smart Meters into Management Systems in Automation. In *Proc. of the 10th IEEE International Workshop on Factory*

Communication Systems (WFCS 2014), 2014.

A. Fernbach, W. Kastner, S. Mätzler, and M. Wollschlaeger. An OPC UA Information Model for Cross-Domain Vertical Integration in Automation Systems. In *Proceedings of the 19th IEEE Conference on Emerging Technologies and Factory Automation (ETFA'14)*, 2014.

W. Kastner, L. Krammer, and A. Fernbach. State of the art in smart homes and buildings. In *Industrial Communication Technology Handbook, Second Edition*, pages 55–1–55–20. CRC Press, 2014.

A. Fernbach, W. Granzer, and W. Kastner. Knowledge Based Building and Energy Management. In *Tagungsband der e-nova 2015, Forschungs- und Studienzentrum Pinkafeld*, 2015.

A. Fernbach and W. Kastner. Gebäudemanagement durch wissensbasierte Systeme. In *7. Jahreskolloquium Kommunikation in der Automation - KommA*, 2016.

A. Fernbach, I. Pelesic, and W. Kastner. Linked Data for Building Management. In *Proc. of the 42nd Annual Conference of the IEEE Industrial Electronics Society (IECON 2016)*, 2016.

A. Fernbach and W. Kastner. Gebäudemanagement durch wissensbasierte Systeme. In *Kommunikation und Bildverarbeitung in der Automation: Ausgewählte Beiträge der Jahreskolloquien KommA und BVAu 2016 Zum 10jährigen Jubiläum des InIT - Institut Für Industrielle Informationstechnik*, Technologien für die intelligente automation, pages 97–106. Springer Vieweg, 2017.

A. Fernbach and W. Kastner. Semi-automated engineering in building automation systems and management integration. In *Industrial Electronics (ISIE), 2017 IEEE 26th International Symposium on*, pages 1528–1534. IEEE, 2017.

D. Schachinger, A. Fernbach, and W. Kastner. Modeling framework for IoT integration of building automation systems. *at - Automatisierungstechnik*, 65(9):630–640, September 2017.

CHAPTER 2

State-of-the-Art

2.1 Knowledge representation

In the history of knowledge representation, a wide range of formalisms has evolved. The major goal in all cases was to fit the field of application it was intended for. In many, but not all cases, a machine interpretable representation of reality (or a small part of the world) is to be created. The formalisms vary in their expressive power and the way logical inferences (i.e., the regulation on how conclusions are drawn) can be carried out. A basic distinction can be made between logic-based languages and non-logic-based ones. In the following, these two categories are sketched by their main representatives in an order of their expressive power [26], [27]. Hereby, the goal is to examine these knowledge representation languages with respect to their suitability to represent information from the (building) automation domain. Also, the non-functional attributes introduced in Section 1.3 play an important role in this course.

2.1.1 Non-logic-based modelling languages

One of the oldest forms of knowledge representation is the *semantic network* [26]. It uses a graph notation consisting of vertices and directed edges to model relations between the vertices. There are no formal restrictions on the structure of such a graph. Though, there are a number of subtypes of semantic networks which actually have restrictions. This very vague, general definition of semantics makes semantic networks in their basic form suitable for intuitive visualisation of knowledge but unsuited for machine processing.

A special form of semantic networks are *taxonomies* where the semantics of specialisation and inheritance is introduced. This way, hierarchically ordered relations can be realised between entities. This allows to establish a categorisation of the entities of interest. This is on one hand a powerful modelling concept but on the other hand - since it is the

only allowed form of relation - a very strong limitation to the expressive power of this modelling language.

An extension to this limited semantics is provided by *thesauri*. They allow additional hierarchical and non-hierarchical relations between entities. These relations may express similarity or associate acronyms and inverses. The ISO standard [28] defines such standard relations between terms.

A *topic map* further increases the expressive power of thesauri by introducing additional semantics to references. It aims at the integration of heterogeneous information resources by means of a standardised solution. Its basic structure is laid down as an ISO/IEC specification [29]. Also an XML syntax is defined [30]. However, no open query language exists to access the information stored in a topic map. Some proprietary query languages processors were developed yet, but without any considerable impact.

Frame-based models follow principles which are known from object oriented modelling. Modelling concepts for classification, generalisation and specialisation, inheritance and instantiation are provided. Class frames have a number of slots associated in order to describe properties of the classes. Hierarchies of class frames reflect a generalisation or specialisation relation. Instance frames are linked to class frames in order to reflect their membership. Additionally, there are ways to include procedural information, i.e., statements to be executed when a slot value is changed.

From the perspective of expressive power, the previously defined requirements on first order integration can be best fulfilled by candidates from non-logic-based modelling languages. Frame-based models provide sufficient modelling power to define type hierarchies of devices, datapoints and topological elements. Type-safety can be assured when these types are instantiated in order to set up a representation of a real-life facility. This supports the creation of syntactically correct instantiation. When following the membership relations from instances back to the corresponding type definitions, valuable meta-information can be gained about the instances. A number of Object Management Group (OMG)[1] standards exists which can be interpreted as frame-based modelling languages. The Unified Modeling Language (UML) [31] class diagram is probably the most popular one. Its notation is loosely based on logic, however there is no formal specification in any logic-based form [27]. Another, yet outmoded one is Common Object Request Broker Architecture (CORBA, also an OMG standard) [17] defining a simple object model. Another object model comparable to the UML class diagram is defined by the OPC Unified Architecture (OPC UA) specification [32]. It additionally provides a predefined but extensible base model comprising a number of general-purpose types. The Open Building Information Xchange (OBIX) [33] object model is even leaner compared to OPC UA but similarly extensible. Both standards use an XML-based language to describe information models. This allows easy, automated transition to other XML-based description languages.

[1] https://www.omg.org

Semantic correctness of knowledge representation in both OPC UA and OBIX can be ensured by a proper design workflow when mappings from automation technologies are created (cf. Section 1.3.1). These mappings are in many cases defined during a standardisation process of community members. Recurring review cycles help to avoid malicious mapping rules. This includes also automatically executed transformation scripts (like Extensive Stylesheet Transformation (XSLT) [34]) which implement these rules. Correct syntax of the XML documents used to describe the knowledge representation can be ensured by automatically checking against XML Schema Definitions (XSDs) [35].

Since frame-based models have no formal semantics, logical reasoning is not applicable here. However, computational complexity to access information on system implementing these standards is low if efficient data structures are used for storage. A usual form to do so is by using hash maps which have a time constant runtime complexity to access an element denoted as $O(1)$ [36].

The learning curve behind information modelling in OPC UA and OBIX has different shapes since the complexity of the base information models defined by the standards highly differs. OPC UA has a rather high-level base model with a great variety of built-in types. The OBIX object model on the contrary is lean and easy to overview. This has also influence on the modelling effort for the individual mappings. Obviously, a modelling goal is easier to reach using a base model consisting of already advanced concepts compared to building upon the very basic OBIX model. To support the designer, a number of standard works exists in literature on modelling in both technologies, e.g., in [37] and [38]. Community support in form of online materials is mainly provided by the standardisation bodies behind OPC UA and OBIX as well as some companies developing software in this context.

With regard to accessing the desired information, a protocol is required which defines the necessary procedures to read and modify data values or to execute functions. Additionally, there is the necessity for security mechanisms to allow confidential, authenticated and authorised communication. UML does not define any form of data access protocol since the purpose of this modelling language solely lies in supporting software and system engineering processes. CORBA, however, provides an abstract message protocol called General Inter-ORB Protocol (GIOP). More precisely, its concrete incarnations allow message transfer on various transport protocols including state-of-the-art security mechanisms. Likewise, the data exchange protocol in OPC UA allows various options on the underlying transport mechanisms and implements security by default. OBIX relies on standard Web services (WSs) like HTTP or Constrained Application Protocol (CoAP) [39] for data transport. To establish secure communication, state-of-the-art Web service security protocols can be used.

Besides the technical properties of OPC UA which are perfectly tailored to automation systems integration and middleware implementations, another aspect speaks for this technology. By now, OPC UA is broadly accepted by the community in automation which means there is tight cooperation between the involved organisations. This results in a number of technology models which are ready to be reused and to build new

solutions upon them. Additionally, support on tools and implementations is remarkable. This significantly eases the scientific work on OPC UA prototypes compared to other integration methods. These arguments partly also apply for the OBIX standard. Its XML-based architecture assures good modelling tool support whereas implementations can simply be built upon standard Web service Software Development Kits (SDKs).

2.1.2 Logic-based languages

The main characteristics of logic-based languages are their formal semantics, the thereof resulting unambiguity and their comprehensibility for machines. Classical logic is a very old form of knowledge representation. Its simplest form, the *propositional logic* has very limited capabilities with regard to its expressiveness but is well suited for automated processing. *First-order logic* is an extension to the two-stated (Boolean) variables a propositional logic formula consists of. Here, predicates and functions on variables as well as quantifiers are introduced which allow to make statements on variables and express the range of validity of these statements. The disadvantages of pure first-order logic formulas are their bad manageability for humans, especially with increasing complexity of the knowledge representation and their general undecidability. This makes automated reasoning in general impossible. A way to overcome these drawbacks is to introduce restrictions on the classic first-order logic which derives new logic families.

Conceptual graphs are a logic-based form of semantic networks and frame-based models. Their visual character facilitates human manageability. Nevertheless, they can be interpreted as a restricted form of first-order logic and therefore enable logical reasoning. There exists an interchange format for conceptual graphs which is one dialect of *Common Logic* defined as an ISO standard [40]. These forms of knowledge representation were superseded by a closely related logic family, the *Description Logic*. Languages from this family are also decidable which makes efficient reasoning possible. Here, knowledge representation is divided in two areas, a Terminological Box (TBox) and an Assertional Box (ABox). The first one defines general domain knowledge (the so-called vocabulary) in form of concepts and roles. Concepts are sets of entities which can be in binary relations to each other, the roles. The ABox can be seen as a real-world description in form of an instantiation of concepts from the TBox. They represent facts from the domain of interest. The famous *KL-ONE* [41] was the first incarnation from the family of description logic [42].

An important service provided by a description logic is to reason about the stored knowledge. On the TBox, reasoning allows to decide Satisfiability (SAT), i.e., the absence of contradictions and determine subsumption relations, i.e., whether one description is more general than another. On assertions stated by the ABox, consistency checks are performed and entailment is determined with respect to descriptions from the TBox. So reasoning fulfils two purposes: to ascertain that a description logic-based knowledge representation is meaningful and to infer knowledge which already implicitly exists, i.e., to make it explicit.

Another powerful feature of description logics is that they allow queries. They formulate concept descriptions (i.e., a class of objects, comparable to the TBox) which a query processor takes as input. It performs instance tests determining if there are individuals which are instances of the query. These queries can be used by applications as an interface to a software system implementing a description logic-based knowledge representation. The latter are called *Knowledge-Based Systems (KBSs)*.

Description logics have reached a wide-ranged prevalence in the IT world in form of the Semantic Web technologies [43] and especially the Web Ontology Language (OWL) [44]. As the name reveals, OWL is designed to describe content on the World Wide Web. The term *ontology* originates in philosophy where it describes a field which aims at finding a description of reality. A common definition reads as follows [27]: "An ontology is an explicit specification of a shared conceptualization that holds in a particular context."

Semantic Web technologies provide the desired powerful methods for knowledge representation. Easy manageability for machines is achieved by their XML-based language. This also facilitates easy transformation from and into other XML-based models like OPC UA or OBIX defines. A big academic community promotes the design of OWL ontologies [45] in various fields of application. A number of them exists describing matters in medicine, biology, pharmacology, physics and chemistry.[2] Two very mature and well-supported open source implementations exist in this area: The ontology design tool Protégé[3] developed by a research group of the Stanford University and Apache Jena[4], a Semantic Web framework. The knowledge stored in the Jena knowledge base can be accessed via the SPARQL Protocol and RDF Query Language (SPARQL) [46], a query language in the sense like described before. Queries and their responses can be transported by HTTP/HTTPS assuring easy integration and state-of-the-art security. Access control mechanisms can be described by ontologies [47] and realised within the knowledge base.

These properties make Semantic Web technologies a well-suited method for second level integration of automation systems. The expressive power of OWL ontologies allows to create comprehensive representations of automation systems, plant topologies, building structure and properties as well as meaningful mapping of runtime data. The scalability of the available knowledge base implementation makes it possible to instantiate representations of large-scale facilities with thousands of datapoints and detailed plant and building models. Structured access to the knowledge base is enabled via a SPARQL endpoint also supporting filtering as well as arithmetical and logical operations on the results. The required security and access control measures are assured.

In ontology design, there exists a state-of-the-art method to evaluate the requirements on an ontology, namely the formulation of competency questions [48]. These formulations reflect the requirements, i.e., it must be possible to answer them by using only the

[2] http://lov.okfn.org/ is an open repository for ontologies
[3] https://protege.stanford.edu/
[4] https://jena.apache.org/

knowledge of the designed ontology. A semi-formal evaluation can be carried out when competency questions are translated to a suitable query language like SPARQL. These queries are fed to a query processor interfacing the respective knowledge base and the results are checked if they contain the required entities.

Regarding the complexity of ontology design and the resulting learning curve, it can be stated that a considerable amount of time needs to be invested to be able to perform state-of-the-art ontology modelling. A basic understanding of description logic and the functioning of reasoners is required. For the most widespread tools and frameworks learning materials like literature and tutorials are broadly available. To have an overview over the plethora of already existing ontologies is also beneficial since reuse of existing work needs to be considered in the highest possible degree.

Efficiency in processing and reasoning on ontologies is evidently related to the algorithms used to solve the problems behind. Besides executing inference rules which make implicitly available information explicit, an OWL reasoner has to check the consistency of the knowledge base. The operation behind that is resolving SAT, i.e., to determine if the iteratively derived instances are a model of the TBox. SAT is an NP-complete problem. The basic procedure for SAT solving is the Davis-Putnam-Logemann-Loveland (DPLL) algorithm [49] which has a worst-case complexity of $O(2^n)$. Huge efforts have been made over decades to improve performance of SAT solvers [50] which resulted in execution times far away from the worst case [51]. Today's OWL reasoners like Pellet [52] have an acceptable performance for knowledge bases of a scale of some ten thousands of classes [53]. On a standard PC, reasoning times for these instances are in the order of magnitude of a minute.

In order to summarise this chapter, Table 2.1 gives a comparative overview on the attributes and capabilities of the knowledge representation forms introduced above. It does not only consider attributes of the considered description language but also properties and features of state-of-the-art software implementations based on these technologies. The previously defined requirements on the knowledge representation and on information exchange are reflected by this table.

Table 2.1: Comparison of knowledge representation languages

	Semantic networks	Taxonomies	Thesauri	Topic maps	Frame-based models				PL	FOL	DL	
					UML	CORBA	OPC UA	OBIX			KL-ONE	OWL
Expressivity for datapoint model	X				X	X	X	X		X	X	X
Expressivity for device model	X				X	X	X	X		X	X	X
Expressivity for topology model	X				X	X	X	X		X	X	X
Protocol for runtime interaction						X	X	X				X
Browse interface						X	X	X				
Query interface												X
Logical reasoning									X	X	X	X
Logical, algebraic operations and filters												X
Security and access control						X	X	X				X
Machine-friendly description language							X	X				X
High scalability							X	X				X
Complex base model					X		X			X	X	X

2.2 Integration technologies

2.2.1 Motivation

Within the context of automation systems integration, *OPC UA* is one of the most important standards supporting WSs. While OPC UA is already well-established in industrial automation systems [54, 55], it gains importance within the building automation domain. OBIX [33] and BACnet/Web Services (BACnet/WS) [56] are integration standards dedicated to building automation systems. These standards can be used to provide a generic view to management clients that need global access to the entire BAS. In order to provide a common interface to different network technologies, integration servers need to be deployed. Visualisation and trending software is a typical field of application where access to process data from management level can be provided via OPC UA, OBIX or BACnet/WS clients. Also, aggregation of alarms and events as well as centralised access to parameters for manual intervention are applicable tasks. BACnet/WS and OBIX follow similar approaches in information representation and transport mechanisms. However, BACnet/WS is out of the focus of this thesis.

2.2.2 OPC Unified Architecture

As a result of the plethora of mutually incompatible standards in automation systems, a standard named *OLE for Process Control (OPC)* was released in 1995. Its goal was to make these different standards and technologies interoperable by means of data representation and data transport. The association which published this standard was called *OPC Foundation*. The principle behind OPC was that each vendor of network components provides dedicated drivers which link the individual network protocols to the OPC Application Program Interface (API). This way, data exchange across multiple technologies is enabled via the uniform interfaces of OPC.

Not only the access to life process data *(Data Access, OPC DA)* was of interest to the OPC Foundation, but also the handling of *Alarms and Events (OPC A&E)* and the access to archived process data *(Historical Data Access, OPC HDA)*. These three parts together form the so-called *classical OPC specifications*.

Microsoft's Component Object Model (COM) and Distributed Component Object Model (DCOM)) were originally used as APIs in the OPC specification. First an advantage, over time COM and DCOM turned out to have significant drawbacks regarding network transparency, security mechanisms and clearly the lock-in to Microsoft Windows. These drawbacks and the limited abilities of representing complex data led to the release of *OPC UA* as a full replacement of the classical OPC specifications. This new standard combines all the features of the previous specifications but replaces the data transport component by WSs and TCP based protocols. Hereby, platform independence is achieved and the implementation of a strong security concept was enabled. Another new component of OPC UA is the so-called *address space*. Here, any kind of complex data can be modelled. The OPC UA specification defines an abstract base model as well as a set of rules called

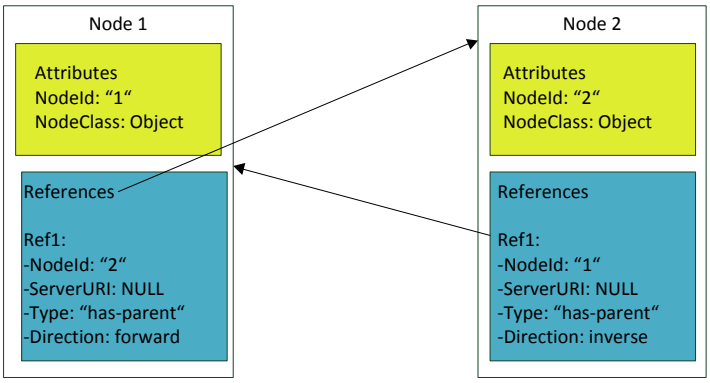

Figure 2.1: Concept of nodes and references [37]

meta model. This meta model follows an object-oriented approach like known from modern programming languages. It allows to derive user-defined information models from the predefined base model.

Information modelling

Contrary to classical OPC which only provides possibilities to represent plain process data, OPC UA [32, 57] supports mechanisms to enrich data with specific semantics. For example, in addition to the measurement value of a sensor an engineering unit and information about the sensor status can be modeled, too. This form of meta data can be interpreted by clients and used by applications to provide additional information related to process data. OPC UA defines the following rules regarding information modelling:

- Information is modelled in form of nodes carrying attributes and references linking the nodes (cf. Figure 2.1).
- Type hierarchies and inheritance are used as object-oriented principles.
- There is no distinction between the exposure of data and type information. The latter is needed by clients to interpret the data which is accessed.
- Information is modelled in form of a network of full-meshed nodes. There is no unique way to model information. Each use case requires a specific manner of modelling.
- The base information model as part of the specification is extensible with regard to defining subtypes of nodes and references between them.
- Information models only exist on OPC UA servers. Clients gain their knowledge about how data is modelled by fetching that information from the server.

The address space contains definitions for basic data types, references, for creating variable types, object types, reference types, methods and further information entities. The basic idea beneath the address space model is the concept of *nodes* and *references* (cf. Figure 2.1). Nodes in OPC UA consist of attributes which give a description of the node and references creating links between nodes. Some attributes are inherent in all node classes, some are specific. Examples of common attributes are the `NodeId` for uniquely identifying the node in the address space, the `BrowseName` which identifies a node when browsing through the address space, and the `DisplayName` attribute containing the name of the node to be displayed in a user interface. The entire list of attributes can be found in Part 3 of [32].

The following built-in instance definition node classes are defined in OPC UA:

- `Object` node class: objects consist of variables, methods, and properties. They are used to model devices or components of the technical process under control, like a temperature controller or a motor controller.
- `Variable` node class: variables must always belong to another node (e.g., an object). The `Value` attribute holds a physical value of a technical process (if it is linked by a `HasComponent` reference) or provides meta information, i.e. characteristics like an engineering unit for the superior node (when referenced by `HasProperty`). In this case, a variable is called *Property*. Properties can neither be of a complex type nor have any subtypes.
- `Method` node class: methods are always referenced to an object. They represent functions that can be called by an OPC UA client (e.g., start and stop routines of a motor controller object).
- `View` node class: in order to reduce the scope of a client accessing an information model on a server, views can be used to make only parts of it visible. Depending on the use case, only the relevant part of the whole model can be made visible to the client.

The built-in type definition node classes are the following:

- `ObjectType` node class: specifies the type of an object. `ObjectTypes` can also be complex or simple where the difference is whether they expose a structure of other nodes beneath them or not. Complex ones can hold other objects, variables, and methods. This allows the engineer to create models of technical devices that reflect the entirety of the relevant device properties.
- `DataType` node class: defines the data type of the value attribute of a variable or variable type. `DataTypes` are organized in a type hierarchy, with the abstract `BaseDataType` on the top. Typical built-in `DataTypes` are `Boolean`, `String` or `Number`. Subtypes of the abstract `Number` DataType are `Integer`, `Float` and `Double`.

- `VariableType` node class: used to define the type of a variable. There are simple `VariableTypes` which only define the semantic and the data type to be used for the value attribute, where complex variable types hold a structure of nodes which enables structuring the variable type into subvalues.

- `ReferenceType` node class: used to specify reference types. References in OPC UA derived from reference types are applied to create a link between two nodes. There are both abstract `ReferenceTypes` and concrete ones. The idea is the same as for `DataTypes`, namely to aggregate common attributes of subtypes within an abstract supertype and generate a more structurised type hierarchy this way. References can either be symmetric or asymmetric, depending on whether they have the same semantic in both directions or not. The `Symmetric` attribute of the `ReferenceType` indicates this property.

Interoperability between devices of different vendors requires a uniform representation of data. In OPC UA, the idea is to define information models (i.e., data representations) for different application domains. Vendors can use these models to expose data of their applications or they can even extend them by their own domain-specific knowledge. Clients do not have to distinguish between different vendors for their functionalities since they all have the same base model exposing data in common. Displaying current process data in a simple, generic user interface, access to historical data or event-driven update of data exposed or signalisation belong to these basic functionalities. If a server provides an information model with functionalities extending these basic ones, clients are able to interpret this more complex data by gathering the additional semantic from the information model. This way, advanced visualisation, more sophisticated computing or automated integration into other systems can be done with data provided by an OPC UA server.

The *Base OPC UA Information Model* is founded on the rules of the meta model. The structure of this part is shown in detail in Figure 2.2. Here the additional specifications known from the classical OPC standard like *Alarms & Conditions (AC)*, *Historical Access (HA)*, *Programs (Prog)* and automation specific *Data Access (DA)* features are included.

Standards published by other organisations use these OPC UA information models and build their own specific ones on top of it. The uppermost layer of information models is formed by vendor specific extensions designed for particular applications using the OPC UA Base, the OPC UA Information Models or other OPC UA based models.

Data transport

OPC UA defines a set of services to exchange data between servers and clients. Contrary to the classic OPC specification, services in OPC UA are defined transport protocol and platform independent which requires an abstract service description (cf. Part 4 of [32]). OPC UA gets along with a very generic and reduced set of services. This is possible since

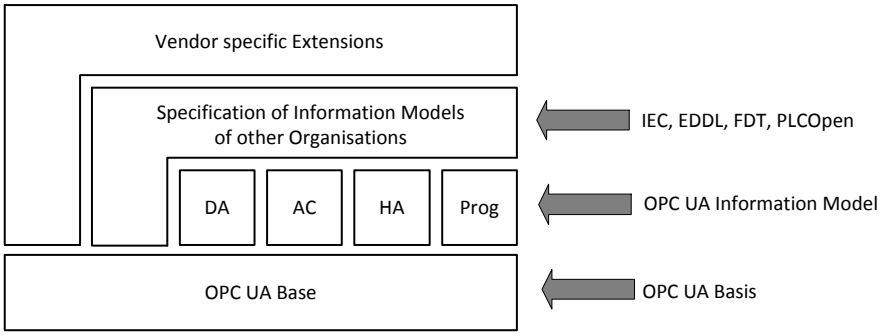

Figure 2.2: OPC UA layered architecture [37]

information is provided by the server address space. There is no need for specialised methods for accessing different types of data or information.

The *Discovery service set* enables an OPC UA client to receive a list of online OPC UA servers from a *discovery server*. To find the node holding the desired data in the information model on the server, the client can directly access it using the `NodeId` or it must browse to the target node starting at a dedicated node called *Entry Point*. The `Browse` service is a representative of the *View service set*. Following the outgoing references, the client reaches the destination node holding the desired information. This is done in a recursive way by calling the `Browse` service for each node on this path. It returns an array of references originating in the node and pointing to a target node. Filtering mechanisms help to reduce the amount of data returned. When the requested node is reached, it can be identified by its `NodeId`.

The *Attribute Service Set* provides access to the attributes of nodes which are uniquely identified by the `AttributeId` and the `NodeId`. These are passed to the service methods as parameters. The most essential services are the `Read` and `Write` service. Depending on the desired kind of access, one of these is called by the client. This is the most common use case to access data. Attributes of an array-type can be accessed element-wise by passing an index argument to the service method. But it is also possible to read or write the entire set or a range of elements of this attribute type as a composite.

2.2.3 Open Building Information Xchange

The OBIX specification [33] is published and maintained by the Organization for the Advancement of Structured Information Standards (OASIS)[5]. The intention behind this standard is to provide Web service-based access to embedded systems and to facilitate Machine-to-Machine (M2M) communication over the World Wide Web (WWW). Hereby, the domain and vendor specific low level protocols used so far in this area shall be superseded. OBIX uses standard Web technologies like XML, JavaScript Object Notation

[5] https://www.oasis-open.org/

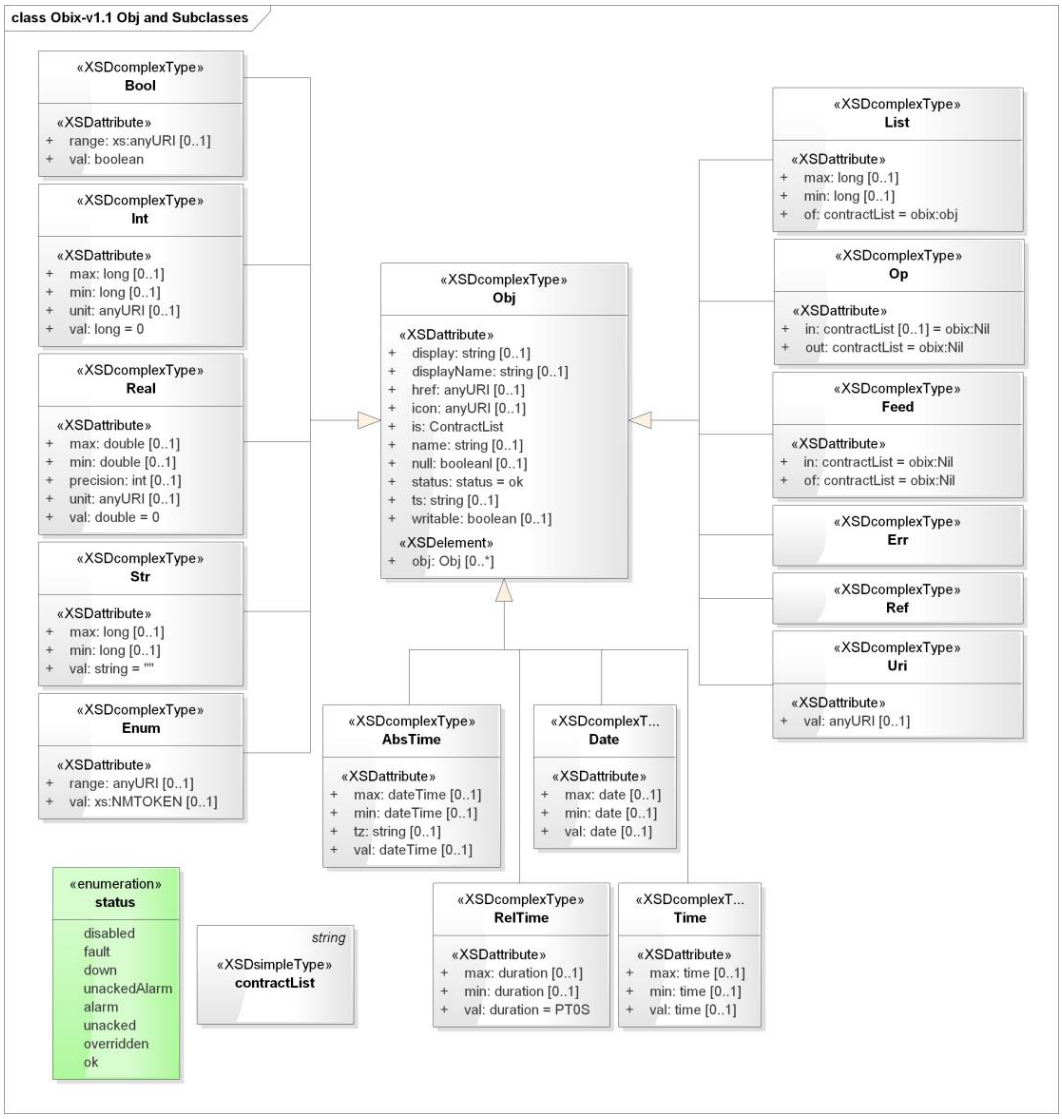

Figure 2.3: OBIX object model [33]

(JSON), URIs for resource identification and data encoding and provides a number of protocol binding options as data transport mechanisms.

Information modelling

For information representation, OBIX defines a simple but extensible object model. Figure 2.3 shows an overview thereof. It defines a number of standard objects to model datapoints of primitive types like `bool`, `int`, `real`, `str` and `enum`. Enrichment by meta-information is achieved by additional attributes for the display name, value range or physical units. References to other objects and lists allow to model complex and nested

object types. In order to represent callable functions on the target system, the concept of operations `op` is available. It provides parameters and return values.

OBIX allows the definition of object templates which are called contracts. They are accessible by the client the same way objects are, but the relation between an object and the contract it implements is modelled by the `is` attribute. Modelling of multiple inheritance and type hierarchies is possible, too. The OBIX core contract library defines a number of standard types, e.g., for representing engineering and temporal units. The `Points` concept is used to model datapoints of systems or devices with OBIX connectivity. Further mechanisms indispensable in automation systems are access to historical data, alarming and subscriptions on value changes. The corresponding OBIX concepts are `History`, `Alarming` and `Watches`. The following listing shows an exemplary OBIX object representing a variable air volume box. Besides a name, a URI (`href`) and a value (`val` of type `real`), it indicates multiple inheritance relations to an `hvac:temperature` and an `hvac:vav` contract (`hvac:` is the notation for a namespace prefix) by means of the `is` attribute. The sub-objects, i.e., XML elements in the following lines, describe additional properties of this object. They are similarly built like the superior object and include attributes for datatype, name, URI and value.

```
<real name="VAV-101" href="/Treitlstr/Floor4/Room22/VAV101/" val="21.0"
is="hvac:temperature hvac:vav">
 <real name="currentTemp" href="spaceTemp/" val="21.0"/>
 <real name="setpoint" href="setpoint/" val="22.0"/>
 <bool name="heatCmd" href="heatCmd/" val="true"/>
 <enum name="sensorType" val="PT1000"/>
</real>
```

Data transport

For data exchange between OBIX servers and clients, four services are defined. The `Read` service is used to transfer the state of objects from server to client. It takes the URI of the object as an argument. By calling the `Write` service, modifications on objects in the server address space can be performed. As arguments, the URI of the object and the new state are passed. If the operation was successful, the server returns the modified object to the client. The `Invoke` service is used to trigger operations on `op` objects. Besides the URI of the `op` object, the input argument object needs to be passed to the service. The server responds with the output object specified in the `op` object if successful or with an `err` object otherwise. The `delete` request is defined to remove an object from the server's address space. The response includes an empty object if successful, or - if not - an `err` object.

The concrete way of implementation of these service requests and responses are called protocol bindings. They all follow the same semantics as described above but rely on different underlying protocols. Currently, the following protocol bindings are defined for OBIX:

- Representational State Transfer (REST) bindings, which include both HTTP [58] and CoAP [39] bindings
- Simple Object Access Protocol (SOAP) [59] bindings
- Websocket [60] bindings

Depending on the platforms in use (which determine the available device resources) and the fields of application, each binding has its own advantages. The CoAP protocol, for instance, is designed for resource-saving operation, whereas the Websocket protocol is on one hand more resource-intensive but on the other hand facilitates event-based, server-triggered communication as used for the OBIX Watch mechanism.

IT security measures like authentication, encryption, access control and user management are not in scope of the OBIX specification. They are dependent of the protocol bindings in use and the overall application context. Data integrity and confidentiality can be provided by standard IT security protocols like the Transport Layer Security Protocol (TLS) [61] or Datagram Transport Layer Security (DTLS) [62]. Access control can be realised by the eXtensible Access Control Markup Language (XACML) [63]. This standard defines how access policies can be formulated and hence describes an architecture of roles and a model how to process access requests according to predefined rules. One middleware integration approach providing an OBIX endpoint and also establishing XACML is the IoTSys project described in [64].

2.3 Semantic Web and ontologies

Today, most of the content in the World Wide Web is designed to be human readable. This makes it hard for machines to meaningfully interpret the information present on the Web. They can essentially follow links from one resources to another but cannot reliably recognise the semantics of resources and resources between them. One way to improve machine processing of the Web is to make the machines smarter, i.e., put effort in artificial intelligence development including natural language recognition and image processing. These are hard problems to solve, but there is an easier way by making the data on the Web more machine-friendly. Information can be represented in a standardised, machine-readable format which allows easy interpreting the meaning of it. This leads to the approach of the Semantic Web.

The Semantic Web can be seen as the next stage in the development of the World Wide Web [65]. The organisation standing behind this technological trend is the World Wide Web Consortium (W3C)[6], a standardisation council which publishes and maintains the specifications all well-known Web standards like HTML, XML or CSS. The Semantic Web vision has its origin in the work of W3C founder Tim Berners Lee [43]. It follows the idea that relations between the resources on the Web need to be established such that

[6] https://www.w3.org/

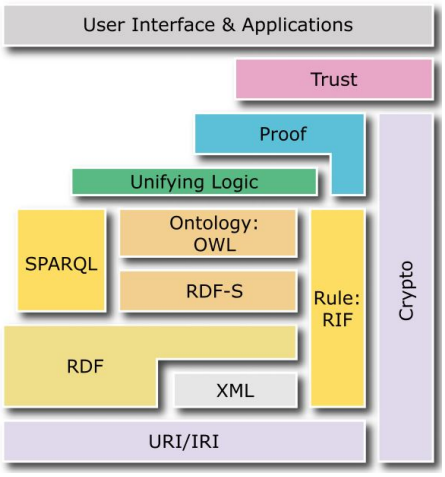

Figure 2.4: Semantic Web stack [66]

they are put in an interpretable context resulting in a "giant global graph" (a term also introduced by Berners Lee) of information. This way, data can easily be interchanged between related but distributed resources. These links between resources need to have a name to add a meaning to the relation and to allow distinction between different kinds of relations. For instance, a link to a person's Web page needs to be differentiated from a link to a person's work address. The way how these mechanisms behind the Semantic Web are established needs to follow distinct rules to ensure interoperability between machines browsing and processing the data exposed. A model needs to be defined on which every participant agrees on. To this aim, a number of standards has been published by the W3C. On one hand, formalisms with different expressiveness capable to create representations of real-world things are defined. On the other hand, languages to perform queries and updates to this distributed knowledge were created. The so-called Semantic Web stack shown in Figure 2.4 illustrates a semantic hierarchy of these concepts. Well-known principles for resource identification (Uniform/Internationalized Resource Identifier - URI/IRI[7] which are set up by Unicode characters) and data encoding (XML) form the basis of it. The Resource Description Framework (RDF), the RDF Schema (RDFS), and the OWL on top define the model of relationship between resources and the format of data interchange. These layers of the Semantic Web stack which are relevant for this work are described in the following sections.

2.3.1 Resource Description Framework

The RDF [67] is a model of subject-predicate-object (S-P-O) triples where the predicate links the subject to the object via a binary relation. Predicates are also-called properties which reflects their semantic of assigning a property to the subject. Figure 2.5 shows an example of an RDF triple where the statement "`Room123 contains`

[7] An IRI is an URI using Unicode as an extended character set

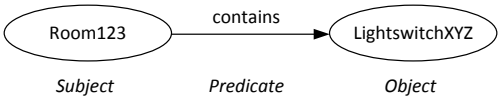

Figure 2.5: RDF triple

`LightswitchXYZ"` is formulated. Subjects, predicates and objects can be IRIs identifying entities in the world to describe. Objects can also be literals which have a datatype (in RDF standard XSD types) assigned to, e.g., `<"true",xsd:boolean>`. When either subjects and objects or both are blank nodes, the statement expresses a relationship but without identifying the involved resources.

In RDF, a set of statements, i.e., triples is called graph, where subjects and objects are vertices and predicates are the edges linking them. Two or more triples can be linked to each other by having the same IRI in their elements. The previous example triple `"Room123 contains LightswitchXYZ"` can be linked to another triple `"LightswitchXYZ hasValue <"true",xsd:boolean>"` resulting in a simple RDF graph. Resources used in one triple need not be located in the same RDF document but can be distributed over the WWW fulfilling the Semantic Web paradigm and leading to a Web of linked data. A set of RDF graphs can this way be connected by one graph referencing resources from other graphs.

As exchange formats, three so-called serialisations for RDF graphs exist. They are logically equivalent but have different properties in automated processing and human readability. The most human-friendly serialisation is the "Turtle family of RDF languages" which in turn is the set of N-Triples, Turtle, TriG and N-Quads. In principle, all Turtle family formats encode triples in a line-based subject-predicate-object form with various extensions. Turtle reduces the verbosity of N-Triples by introducing shortcuts. TriG extends Turtle by the ability to represent multiple graphs and also allows naming them. N-Quads extends N-Triples towards expressing named graphs but keeping the verbosity of N-Triples. RDF/XML was the first serialisation format defined providing an XML syntax. Since XML is a well-established Web standard where parser implementations exist in big variety, RDF/XML is a versatile M2M exchange format. The same holds for the The JSON Data Interchange Format [68], [69] serialisation of RDF, the JSON-LD syntax. JSON is, like XML, also a very common M2M exchange format, but because of its compact syntax with much better human readability.

2.3.2 RDF Schema

The RDFS [70] provides a simple vocabulary for modelling data in RDF. This introduces the term *ontology* which is defined as a formal model of the domain of interest. RDFS extends the limited set of built-in modelling capabilities of RDF. This way, more fine-grained statements can be made about data in RDF. RDFS introduces mechanisms for classifica-

tion, generalisation and specialisation of resources. The most important modelling constructs to this aim are `rdfs:Resource`, `rdfs:Class`, `rdf:Property`, `rdf:type`, `rdfs:subClassOf`, `rdfs:subPropertyOf`. Using these concepts, classes and hierarchies of both resources and properties can be defined. Using the `rdf:type` property, statements about the membership of data to classes can be made.

Another feature of RDFS is the ability to define constraints on the domain (`rdfs:domain`) and the range (`rdfs:range`) of properties. The domain is the set of subjects a property can be used on by definition whereas the range is the set of objects which subjects from the domain can be linked to by the respective property.

RDFS this way allows to define models of low complexity on data to be modelled in RDF. This introduces possibilities to perform consistency checks (i.e., to automatically verify whether an RDFS model is fulfilled or violated by an RDF dataset) and logical deductions (inferences) on data.

2.3.3 Web Ontology Language

Considering again the Semantic Web stack in Figure 2.4, the OWL [44] (the current version is OWL 2) is the next upper layer over RDFS. This reflects the increased expressive power compared to RDF and RDFS. Models in these languages show a limited expressiveness which is in fact of practical relevance when representing a complex world. To overcome these lacks, OWL refines the modelling capabilities mainly with regard to properties. There is a distinction between properties between resources which are called *ObjectProperties* and properties between resources and literals, called *DataProperties*. For ObjectProperties, additional attributes are defined which are well known from binary relations over sets, like symmetry, transitivity, reflexivity or transitiveness. Also constraints on the existence (i.e., quantifiers) and the cardinality of ObjectProperties can be stated. Other concepts known from set theory used in OWL are *union*, *intersection*, *complements* and *disjointness*.

From a syntactical point of view, in OWL 2 there exist three different categories. *Entities*, which are identified by IRIs, form the basic elements of an ontology. Classes, properties and individuals (instances of classes) belong to this set of primitives. *Expressions* are logical statements about entities. They describe Boolean connectives, set theoretical statements and cardinality restrictions on properties. The *axioms* in an OWL 2 ontology state assertions which are valid in the domain to be described. They include relations between classes and properties (equivalence, disjointness, subset) or domain/range assertions on ObjectProperties and DataProperties.

In order to provide adequate modelling capabilities - and as a consequence thereof also reduced computational complexity - OWL 2 defines dialects with a reduced level of expressiveness. The *OWL Direct Semantics* directly assigns meaning to the ontology which results in a semantics equivalent to a description logic, provided that a few restrictions are fulfilled. Therefore, this semantics is informally called OWL 2 DL. The resulting description logic is a fragment of first order logic for which algorithms with

reasonable runtime complexity exist. The complexity of OWL Direct Semantics is further reduced by the three profiles defined in OWL 2: *OWL 2 EL*, *OWL 2 QL*, *OWL 2 RL*. Adapted for different application scenarios, the reduced syntax of these profiles further increases performance and scalability. *OWL 2 RDF-Based Semantics*, the counterpart of OWL Direct Semantics, does not require any syntax restrictions and is therefore informally called OWL 2 Full. A set of terms for mapping OWL 2 ontologies to RDF graphs is specified under [71] as part of the W3C OWL 2 document suite.

2.3.4 Knowledge-based systems

When it comes to computer programs which solve problems by operating on an extensive knowledge base, the term *KBS* has been evolved [72], [73]. KBSs originate in the field of artificial intelligence with the intended goal to assist or even replace human experts in the work on complex tasks. They were therefore often called expert systems.

A characteristic property of a KBS is the separation of knowledge representation about the domain of interest and the knowledge processing. For knowledge representation an appropriate language (e.g., RDF or OWL) needs to be chosen. The main processing tasks in a KBS are logical inferences (also-called reasoning) and the answering of queries. This separation makes KBSs flexible with respect to the knowledge domain since the processing mechanisms are generic and independent on the knowledge base. Also modifications and extensions on a KB can easily be accomplished. Openness for adoptions is in this sense a key feature since knowledge is in most cases incomplete, containing errors and changing over time. Logical inconsistencies can be detected by reasoners.

The vocabulary of a description logic-based knowledge base comprises three kinds of elements: concept names, role names and individuals. In OWL jargon, concept names conform to class names, and role names are called properties. Individuals are constants representing a part of the domain to be described. Using constructors for concepts and roles, more complex relations can be defined. These constructors include Boolean or set-theoretical expressions, as well as quantifiers (universal or existential). The set of available constructors differs depending on the description logic in use. In the TBox of a knowledge base, terminological axioms are defined which state how concepts and roles are related to each other. Two main kinds of axioms exist: definitions where a concept is defined equal with an expression of concepts, and roles as well as general concept inclusions (specialisations) which describe a subset relation between expressions of concepts and roles. The ABox of a knowledge base is a set of concept membership assertions, i.e., statements assigning individuals to concepts and roles. Therefore, the ABox can also be seen as a set of facts which are known to be true in the domain of interest.

Figure 2.6 shows these concepts by means of an example using the syntax usual in literature. The TBox on the left side contains a simple type hierarchy consisting of an `eg:BuildingEquipment` class and two subclasses thereof, `eg:LightSwitch` and `eg:TemperatureSensor`. The subclass relations are expressed by the `rdfs:sub-`

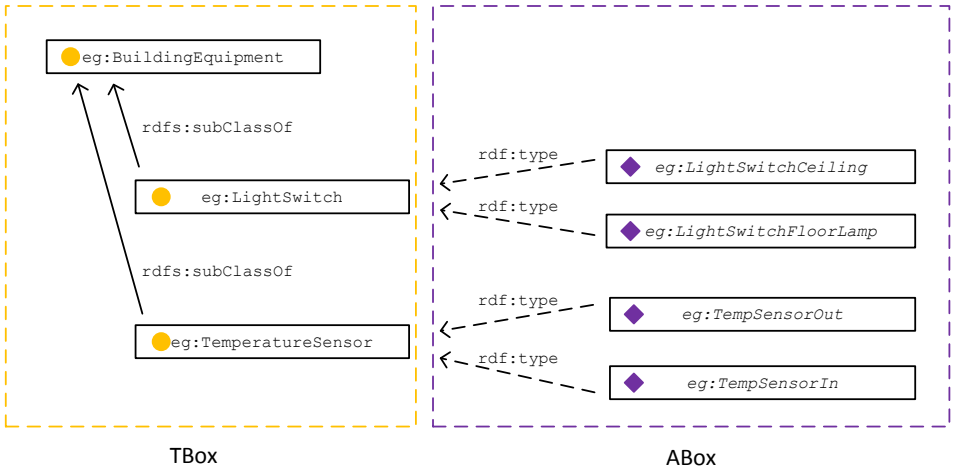

Figure 2.6: Knowledge base example

ClassOf property. Each subclass as two associated instances `eg:LightSwitchCeiling`, `eg:LightSwitchFloorLamp` as well as `eg:TempSensorOut` and `eg:TempSensorIn`. The `rdf:type` properties indicates the membership. Together with the instances, they constitute the ABox of the knowledge base.

2.3.5 SPARQL Protocol and RDF Query Language

Typically, knowledge bases provide a way to access their triple store by means of a query language. In the simplest case, such a query language defines subject-predicate-object (SPO) queries which consist of triples where on each position either a resource name (URI) or a variable ("?") is specified. This way, search patterns are defined which are matched against the graph as which the content of the triple store can be seen. The SPO query `?subclasses rdfs:subClassOf eg:BuildingEquipment` would for example return all subclasses of the class `eg:BuildingEquipment`.

A query language which is defined in form of a W3C recommendation is SPARQL [46]. The specification not solely covers the query language but also a protocol for transportation of the result data. A number of different encodings for the query results allows an adaptation to the receiving application. SPARQL can be seen as an extension to SPO queries with additional features for data filtering, performing arithmetic and Boolean operations on result data and string manipulations. Its syntax has similarities with the database query language SQL. The following query returns all subclasses of the class `eg:BuildingEquipment` and also the instances of these subclasses:

```
SELECT ?subclasses ?instances

WHERE {?subclasses rdfs:subClassOf eg:BuildingEquipment.
           ?instances rdf:type ?subclasses}
```

The lines in the `WHERE` clause form a logical conjunction of statements where multiple occurrences of one variable are always bound to the same value. Modifiers for optional existence of patterns or unification of search results bring a high level of freedom. This way, comprehensive graph search patterns can be created.

Considering the knowledge base example in Figure 2.6, this query would result in a dataset including the classes `eg:LightSwitch` and `eg:TemperatureSensor` as well as their associated instances `eg:LightSwitchCeiling`, `eg:LightSwitchFloorLamp`, `eg:TempSensorOut` and `eg:TempSensorIn`.

In order to perform modifications on RDF triple stores, the so-called SPARQL Update language provides means to insert and to delete data. Similar to the query language, search patterns can be defined which define the location in the graph where items are inserted or deleted.

The SPARQL protocol for RDF defines transport mechanisms for queries and result data to SPARQL endpoints via HTTP services. Using HTTP GET, the query can be passed as an argument of the endpoint's URI. The HTTP POST service allows to include a message body in the request which is especially necessary for SPARQL Update requests. As formats for data encoding, there is the choice between XML, JSON or CSV/TSV or RDF/XML in case a complete graph needs to be transferred.

2.4 Engineering support concepts

The design of building automation systems is a mainly manually carried-out process. Different planning offices work on different trades (subdomains of Building Automation Systems (BASs) like lighting, Heating, Ventilation and Air Conditioning (HVAC), primary plants, room automation) using different technologies. Each technology provides its own engineering tool or even manufacturer-specific engineering tools within one technology.

Efficiency in planning and engineering can be increased by a number of measures. One is reducing error sources - especially at the interfaces between these trades. The planning and engineering workflow can be improved by avoiding overlapping and multiply carried out planning work and by the automatisation of repetitive manual work. This includes clearing the deficiency of entering the same information recurrently in different phases of the planning and engineering workflow. Also the use of libraries for standard design patterns can save time-consuming manual work by just instantiating these templates for, e.g., each office room. To overcome these current weaknesses, a number of ideas exists in scientific literature as well as in the industry. These approaches are briefly described in the following. The focus hereby lies on the building automation domain.

One key concept which can facilitate the engineering process of automation systems is the use of libraries of templates for recurring use cases as well as repositories of devices. One standard that goes in this direction on a rather abstract level is the VDI 3813-2 [74]. It defines hardware-independent functions of room control in form of function blocks with informative character. The function blocks have inputs and outputs with associated,

normative datatypes. By connecting these function blocks, the intended room control can be realised. The datatypes of inputs and outputs to be connected hereby have to be considered, i.e., only ports with identical datatypes are compatible. The defined functions are grouped in the categories *sensor functions*, *actuator functions*, *operator and display functions* and *application functions*. Examples for sensor and actuator functions are *brightness measurement* and *sunshade actuator*, respectively. A function from the operator and display category would be *signal presence* whereas *constant-light control* is categorised as an *application function*.

Originating in the industrial automation domain, the *Computer Aided Engineering Exchange (CAEX)* (IEC 62424) [75] is a meta model which is however rather flexible regarding the domain specific models which can be created by it. The IEC 62424 defines four modelling elements: the *Interface Library* where flow of materials, energy or information can be defined, the *Role Library*, which is used to describe functions, the *Unit Library* to model system units and the *Project Hierarchy* element which is used to structure a plant.

2.4.1 Building information modelling

To the aim to define a digital representation of a facility, its physical and functional characteristics which is shared between the involved parties of a building life cycle, the concept of Building Information Modelling (BIM) has evolved. It acts as an information base throughout all phases of a building's life starting during its design phase and continuously growing in the following [76]. BIM encompasses all information generated during and necessary for planning, bidding, construction, operation, refurbishment and deconstruction. It does not only include geometrical (3D CAD) data but also physical properties of the structure, parametric objects for Mechanical, Electrical and Plumbing (MEP) installation as well as building services. It hereby facilitates the cooperation between trades involved in a building's life cycle. For facility management, BIM is particularly beneficial. A Building Management System (BMS) which can make use of a BIM can effectively put datapoints, alarms, events and associated hardware in a spatial and topological context. Also visualisations which are augmented with state information on building services can be generated automatically. This provides more comprehensive information about the state of building services to the operator personnel and allows them to react on incidents more effectively.

Besides the Industry Foundation Classes (IFC) [77], one popular open BIM standard in the Architecture, Engineering and Construction (AEC) industry is the Green Building XML (gbXML) schema [78]. It is designed as a unified data exchange format between CAD and design tools and tools for energy analysis in order to establish interoperability. Leading software products like AutoCAD Architecture and EnergyPlus from Autodesk or Bentley Architecture and Bentley AECOsim Energy Simulator provide support for gbXML. The schema defines a great variety of parameters concerning the building geometry, its architecture and physical properties. Attributes are hierarchically organised and provide attributes for sites, buildings, storeys, materials, rooms, surfaces and openings like doors

and windows. Structural information is enriched with properties building physics, e.g., albedo, glaze, emittance and U-value. An XSD formalises and explains all available parameters in gbXML and ensures compatibility between the different software tools. The current version of the gbXML schema is 6.01 and has been released in November 2015.

2.4.2 Reference designation systems

The IEC 81346-1 [79] defines principles for structuring objects, i.e., components in plants and buildings. Following these structures, ISO/TS 81346-3 [80] further defines rules how to form reference designations in order to uniquely identify an object within a plant or a building. This results in a systematic naming convention providing information about an embedded object's functionality, location and product related aspect. The benefits of such a reference designation system are the easy identification of systems and their parts and the use of a common language for aligning information about a whole plant containing of an enclosing building structure and the technical equipment. A reference designation describes one up to usually three of the previously called aspects. However, there also exists the possibility to refer to user-defined aspects. Syntactically, each one is signed by a prefix, "=" for the functional aspect, "-" for the product aspect and "+" for the local aspect.

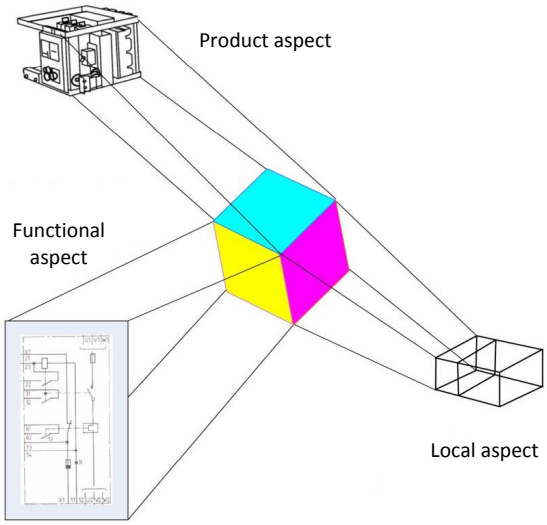

Figure 2.7: Three aspects of an IEC 81346 object (adapted from [79])

Figure 2.7 illustrates these three aspects of a component. Standardised identification letters following the prefixes are used to classify the intended application of an object. The location can be signed by an individual systematic. Considering a motor (M01) driving a pump (G01) of a cooling system (E03) located in room number 4 of building A, the reference designation would look like the following: `-E03-G01-M01+A+4` where the hierarchical structure is represented in a top-down manner.

2.5 Related scientific work

2.5.1 Integration by OPC UA and OBIX

A method how to integrate field devices of various technologies by means of OPC UA is described in [81]. The introduced information models include both concepts for proper runtime data interpretation as well as for tailored Graphical User Interface (GUI) instantiation. The models are based on the the device integration technologies Electronic Device Description Language (EDDL) and Field Device Tool (FDT). For the successor of these technologies, called Field Device Integration (FDI) [82], in the meanwhile an OPC UA companion specification exists.

Another publication from the field of industrial automation [55] combines the information models of ISA-88 [83] and ISA-95 [84] and defines a mapping to OPC UA. The goal is to facilitate the integration of production systems to the Manufacturing Execution Systems (MESs) and Enterprise Resource Planning (ERP) levels. From the efforts undertaken in this direction, another OPC UA companion specification arose [85]. A further publication from this field [86] proposes an approach of an OPC UA interface to ISA-88 control models in the context of the Programmable Logic Controllers (PLCs) programming languages defined in the IEC 61131-3 [87].

Also for warehouse and logistics systems integration, OPC UA has already been considered [54]. An attempt presented in [88] aims at bridging OPC UA and Semantic Web technologies in the domain of industrial automation. This two-stepped approach uses OPC UA to access diagnosis data from sensors placed in a production line and integrates it into a semantic access layer. This middleware contains device and interface descriptions modelled in OWL which allows to perform reasoning and executing SPARQL queries on the provided information.

Also like in practice, OBIX has a low appearance in academia. One integration approach using OBIX is presented in [89], where a mapping of the KNX interworking model is created. This includes a representation of the KNX datapoint model, group communication services and a way for discovering KNX devices. An implementation of an OBIX Web service gateway shows the feasibility of the concepts presented. In [90], [91], [92] and [93], a multi-protocol gateway approach for building automation technologies is proposed, where OBIX is used as an application protocol of the resulting IPv6 [94] Web service endpoint. Additionally, protocol bindings of OBIX to the Efficient XML Interchange (EXI) [95] and the CoAP [39] are defined in this work, which were later included into the OBIX specification. Also addressing Web service communication of Internet of Things (IoT) [96] devices in building automation, the work presented in [97] refers to OBIX as a suitable protocol in this field. In [98], the proposed building monitoring platform uses OBIX as one way to access monitoring data collected from various sources. Additionally to live datapoint values, also historical access is provided and alarms are propagated via the OBIX interface.

2.5.2 Integration by Semantic Web technologies

An approach of modelling device descriptions from the building automation domain by means of ontologies is presented in [99]. The goal is to create a device repository where devices can be retrieved and selected according to the intended usage. The model behind this repository comprises a multi-layered ontology covering a wide range of abstraction levels. The four layers are constituted like the following: generic building automation functions and interfaces, platform-specific properties like controller hardware, datapoint type definitions and manufacturer-specific device descriptions. An RDF triples store is filled with the designed model and a set of instances of concrete devices. Via SPARQL queries, the user is enabled to retrieve devices depending on their properties. The device repository can be seen as an assistance during system engineering where based on requirement specifications device selection including interoperability management is considerably facilitated. However, due to its focus on engineering, no runtime data integration is considered by this approach.

The work in [100] further enhances the device description ontology from [99]. For reflecting the building structure where devices are embedded in, the data model of the IFC building information BIM (i.e., the building structure) is used partly. For modelling functional aspects, concepts defined in the VDI 3813 [74] are used. A design template ontology is introduced which reflects the fact that many design patterns reoccur not only in one installation but also in general in this field. Having a set of these predefined design patterns available is a prerequisite to self-commissioning of devices. An ontology for data access, i.e., to access runtime data is also provided conceptually but the way how live data can be retrieved from automation components remains unclear, however. An approach which operates on a similar information base but aims on a different goal is presented in [101]. The authors use a knowledge base on building automation systems and building information models to reason on fault propagation in this context. Therefore, this knowledge is enriched with causal relations between components and datapoints in such a system. These causalities can be automatically derived from a set of defined rules. Two models of fault propagation are described by the authors following these causalities either in forward or (indirectly) in backwards direction. However, it is still an open issue to integrate additional quantitative methods such that the proposed approach is able to produce reliable outputs.

Another work which is solely dedicated to a mapping of IFC to the ontology domain is presented in [102]. In the meanwhile, the resulting ontology from this work, the so-called IfcOWL ontology, is of considerable size (around 20.000 axioms). It is maintained by the buildingSMART Organisation[8] which is also responsible for the IFC standard. IfcOWL has evolved to a buildingSMART candidate standard which can be seen as a status before becoming an international standard. The very mature open source project BIMserver[9] also provides support for IfcOWL.

[8] https://www.buildingsmart.org
[9] http://bimserver.org/

The Smart Appliances REFerence (SAREF) ontology[10] [103] has a focus on devices and their functions and has a modular architecture. It also defines services which can be provided by devices and are discoverable by others. Also energy aspects are considered. It sets up the basis for a number of other ontologies like the Brick ontology [104]. This "schema for metadata in buildings" encompasses devices (equipment), locations and datapoints. This way, a building topology equipped with automation systems and their datapoints can be modelled. Automation systems can be composed of various subsystems. It is also possible to express control flows from one device to another and media flows like air or liquids in duct systems. On the project homepage[11], the authors not only provide the ontology but also a number of example instances representing existing buildings. In [105], an ontology is defined which aims at establishing a generic application model for building automation systems. A function block model is created covering most common applications. Additionally to the operation phase, also engineering is considered by this approach.

In literature, there exists a number of knowledge-driven approaches for building energy management which base on ontologies. In these works, rather lean ontologies are used for modelling the building and automation system equipment context. Often, inference rules are used to trigger the required actions depending on the current state of the systems. In [106], the DogOnt ontology [107] is used where different ways of instantiating a knowledge base are presented. The manual way requires an expert to transfer information from CAD drawings and planning documents to the knowledge-based domain. In a semi-automatic way, 2D-CAD drawings are interpreted by an algorithm which then instantiates the knowledge base accordingly. Another semi-automatic procedure uses data-mining on energy consumption values to classify consumers, to instantiate the corresponding OWL classes and for generating SWRL rules. [108] follows a manual procedure of instantiating an ontology covering equipment, functions, datapoints and architectural aspects. The resulting knowledge base is embedded into an architecture also comprising a monitoring system for datapoint values and components with analysis and decision functionalities. Inference rules are used to describe the context and causal relations of control actions. In [109], an ontology is defined which aims at structuring the information sources involved in building monitoring. This includes indoor and environmental conditions, control systems, energy flows and inhabitant's behaviour. Utilising this ontology is proposed to facilitate the realisation of multiple applications like facility management and diagnostics by providing a unified description of the underlying services. A more thorough elaboration of this topic is given in [110]. A research project on a modular building management platform with interfaces for multiple protocols for data acquisition and processing tools is described in [98].

In [111], an approach to establish semantic interfaces between components in factory automation systems is introduced. The authors use OWL to describe these interfaces and SPARQL queries to make them accessible for an overlying orchestration engine. An

[10] https://sites.google.com/site/smartappliancesproject/ontologies/reference-ontology
[11] http://brickschema.org/

ontology reflects the required domain knowledge about the equipment, the product to be produced and the production process.

A publication which concerns formal verification of ontological models is presented in [112]. The work focuses on ontology-based descriptions of field devices which shall be checked for consistency of the modelling, completeness with regard to the specification and correctness within constraints given by the domain of interest. Under the open-world-assumption which is inherent to ontologies and the use of standard OWL reasoners, the latter two attributes are impossible to be validated. To this aim, so-called integrity constraints are defined which circumvent the open-world-assumption. Hereby, a validator consisting of an OWL reasoner operating on these constraints is able to perform the desired checks.

2.5.3 Engineering support

A research project where building information models and Semantic Web technologies are used to support the refurbishment of buildings towards energy efficiency is called SEMERGY [113]. Besides building information, multiple further information sources are gathered, like preferences of the building owner, climate data, legal constraints and available subsidies. This information is mapped to a knowledge base and further processed by a reasoner. As an output, the user gets a suggestion on the most suitable refurbishment solution.

An example from the factory automation domain regarding model-driven engineering is presented in [114]. Taking UML models as an input, a code generator creates the structure of IEC 61131-3 software projects which are common in PLC programming. A point of criticism is the still unsolved issue of automatically back-propagating changes on the code to the UML model. In [115], an OPC UA server is used to manage engineering information provided in the CAEX format. Clients are enabled to create instances on the server following the plant structure defined in CAEX. This process image is intended to be used for monitoring purposes. Additionally, a visualisation interface is automatically generated thereof.

The work presented in [116] also builds upon the CAEX standard and aims to establish interoperability between Computer Aided Engineering (CAE) systems of different building automation technologies used in the HVAC or lighting trade. Therefore, it uses the CAEX model and defines a mapping to the building automation domain. This unified engineering format enables the use of a technology-independent engineering tool to supersede the current situation of almost each technology having its own tool.

In [117], an automated design approach for control applications based on IEC 61499 [118] is presented. Therefore, ontological representations of the plant to be controlled as well as an IEC 61499 function block ontology is designed. A set of rules, written in an extension of SWRL [119], is used to perform a transformation from these two ontological representations to a control application program in form of an OWL knowledge base.

The work in [120] initially describes the device ontology used in [99]. The paper further introduces a set of SWRL rules to deduct interoperability between devices regarding the communication medium, functional profiles and I/O semantics. This approach is further proceeded by the interoperability evaluation model presented in [121]. Here, a multi-level hierarchy model is defined which organises different levels of interoperability between automation devices. This model is applied on a dataset of LonWorks devices showing that the search space for automated engineering approaches is drastically reduced.

In [122], [123] and [124], an automated design process comprising three phases of different abstraction is illustrated. For this purpose, a requirement ontology is designed and utilised which allows to describe room-based engineering patterns, highly abstract VDI 3813-2 Functional Blocks (FBs) down to technology-specific functional profiles. The transitions between the design phases are sketched to be carried out automatically by means of generative programming and evolutionary algorithms. For the final mapping of the detailed design to specific automation devices, SPARQL queries are issued to an OWL-based semantic device repository. [125] also addresses the ontology-based automated design concept and additionally describes the evolutionary approaches for solution synthesis in detail. Quantitative results of the evolutionary optimisation process are provided as an evaluation of the work. [126] summarises the publications from these fields and elaborates the whole workflow in detail. The resulting prototype of a building automation design tool is also published on the Web[12].

The work in [127] describes a simple three-level ontology with increasing level of abstraction including functional descriptions, profiles and device descriptions. It is instantiated by a "generator" (generative programming) using a Domain Specific Language (DSL) as a specification. A knowledge-based device repository contains abstract device information which can be automatically matched with the generated abstract instance. Here, multiple degrees of freedom occur which are handled by evolutionary algorithms. An ontology which aims at facilitating the automated deployment of IoT devices is proposed in [128]. It bases on well-established vocabularies like defined in the Semantic Sensor Network Ontology [129].

In [130], an ontology-based specification is used to automatically interconnect services (e.g., corresponding to a light switch an the according actuator) in a smart home application. These services come as OSGi[13] bundles with well-defined interfaces. These interfaces are mapped to OWL constructs which provide the necessary semantic description in order to enable the automated matching. [131] also concerns the automatic configuration in the context of smart homes but by building upon multi-agent systems. A number of these agents is defined where each one manages a distinct functionality. Depending on the intended use case, the available building services - i.e., the output of a discovery mechanism - are utilised.

A semi-automated datapoint mapping from BMSs to energy management systems is

[12] http://www.auteras.de/
[13] https://www.osgi.org/

presented in [132]. The approach used therefore is matching textual labels stored in a dictionary of terms from the building automation domain. A linguistic similarity computation method is applied on these dictionary entries. This way, a matching of terms from the BMS to terms of the energy management system is achieved which corresponds to the desired datapoint mapping.

An attempt to reduce the engineering effort for field device development is made in [133]. A rapid-prototyping method is proposed which bases on a generic device model which is configurable by means of an XML file. This configuration has a modular setup and is aligned to standardised device profiles. A tool chain is developed which semi-automatically generates device configurations based on these profiles.

A thorough survey on the most common system and device modelling approaches covering their whole life cycle is given in [134]. It includes a quantitative comparison of the coverage of information requirements on components during the phases of a systems life. Also, the different requirements between the application domains in focus, i.e., building automation, factory automation and sensor Web, are regarded.

CHAPTER 3

First level integration using OPC Unified Architecture

This chapter is based on the contributions of the author to the following publications:

A. Fernbach, W. Granzer, and W. Kastner, "Interoperability at the Management Level of Building Automation Systems: A Case Study for BACnet and OPC UA," *Proc. of 16th IEEE Conference on Emerging Technologies and Factory Automation (ETFA '11)*, Septempber 2011.

A. Fernbach, W. Granzer, W. Kastner, and P. Furtak, "Mapping ETS4 Project Structure to OPC UA using ETS4 XML Export," *KNX Scientific Conference*, Nov. 2012.

A. Fernbach and W. Kastner, "Integration of Smart Meters into Management Systems in Automation," *Proc. of the 10th IEEE International Workshop on Factory Communication Systems (WFCS 2014)*, May 2014.

A. Fernbach, W. Kastner, S. Mätzler, and M. Wollschlaeger, "An OPC UA Information Model for Cross-Domain Vertical Integration in Automation Systems," *Proc. IEEE 19th Conference on Emerging Technologies Factory Automation (ETFA'14)*, Sep. 2014.

The work presented in this chapter has been carried out in the context of the research projects "Web-based Communication in Automation" (WebCom) and "Information Modeling in Automation" (iModelA) funded by the Austrian Research Promotion Agency.

The development of proper concepts to provide interoperability between different network standards in Building Automation Systems (BASs) is a highly estimated field of research. Since IP based networks are commonly used at the management level of today's BASs, one promising way of communication at this level is the use of Web services (WSs)

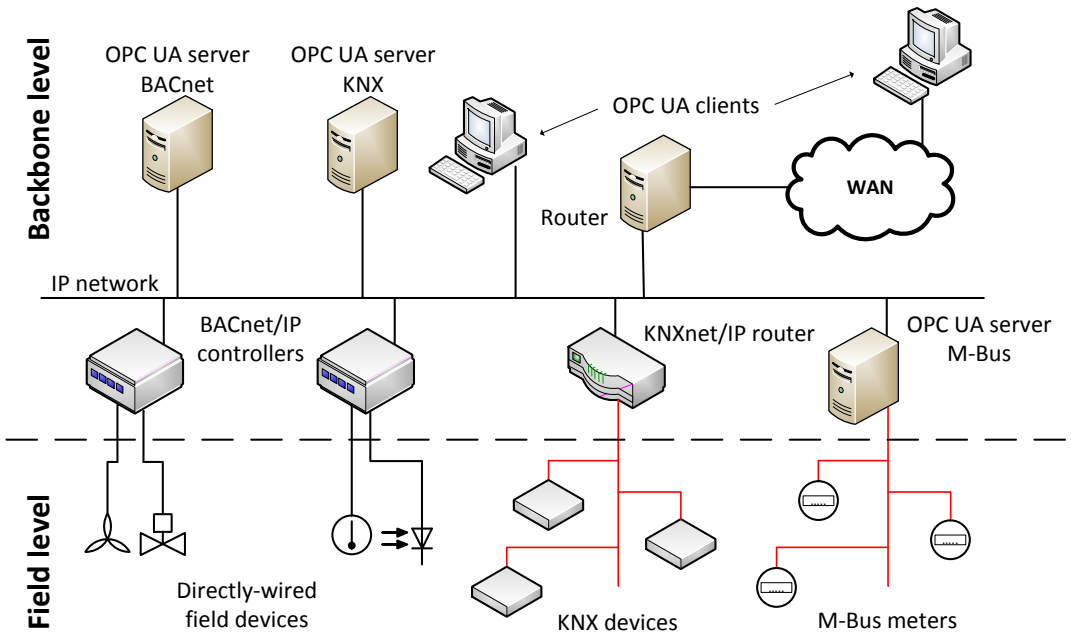

Figure 3.1: Building automation system integration by OPC UA

[13]. WSs have the advantage that they provide platform- and programming language independence. Based on the exchange of messages, WSs follow the Service Oriented Architecture (SOA) paradigm. This enables devices to exchange data independently from the underlying network technologies. Within this context, OPC Unified Architecture (OPC UA) is one of the most important standards supporting WSs. While OPC UA is already well-established in industrial automation systems [54, 55], it slowly gains importance within the building automation domain. The integration standards Open Building Information Xchange (OBIX) [33] and BACnet/Web Services (BACnet/WS) [11] follow a similar intention than OPC UA. Although they are dedicated to the building automation domain, they are even less widespread in this area than OPC UA.

OPC UA emerged as one of the most promising ways to create a unified view on the underlying process data including meta information as well. So as to know how process information is represented and addressed in a certain automation network, OPC UA information models need to be defined. Instances of those models can be used to provide a generic view to management clients which need global access an entire building automation installation. However, to be able to expose live datapoint values this way, interfaces to the underlying technologies are required.

In this thesis, information modelling in OPC UA is the chosen method of knowledge representation for first level integration. Therefore, the following sections present approaches how representatives of the most common building automation technologies, i.e., BACnet,

KNX and Meter Bus (M-Bus) can be integrated into the management level by means of OPC UA. A possible BAS architecture where integration by OPC UA is realised is shown in Figure 3.1. It represents a typical situation where multiple technologies are deployed. An IP-based backbone network interconnects the various components and fieldbus networks. The BACnet devices in this example natively have IP connectivity whereas a KNX bus is integrated via a KNXnet/IP router. This allows the corresponding OPC UA servers to access data from these device on IP level. For the integration of the M-Bus installation, the OPC UA server directly interfaces the bus via an intermediate protocol converter (e.g., M-Bus to EIA RS-232).

In order to provide the necessary technical background, the chosen representatives are briefly introduced in the following Section 3.1. Subsequently, the first level integration approaches of these technologies are described: M-Bus (Section 3.2), followed by KNX (Section 3.3) and BACnet (Section 3.4). Expanding the focus to the industrial automation, the last Section 3.5 presents a common information model creating a unified view on systems consisting of technologies from both domains. To keep the big picture in mind, it is once again pointed to Figure 1.5 which illustrates the context of the technology mappings carried out here.

3.1 State-of-the-art building automation technologies

3.1.1 Meter Bus

The M-Bus [135] is a widespread communication protocol, especially in European countries, which enables the remote readout of smart meters in consumer and industry applications. It has been laid down as a part of the European standard EN 13757 [7], [136]. Smart meters are used to measure the consumption of electricity, gas, heat and water, to mention a few examples [137]. The M-Bus Usergroup[1], which publishes and maintains the standard, mentions further applications like alarm systems, lighting and heating controlling. The protocol can be seen as a fieldbus with a typical bandwidth of 300 to 14400 bps. For communication media either a two-wired twisted pair line (also providing link power) or radio communication can be used. The master-slave communication paradigm applied in M-Bus ensures central media allocation. Metering devices act as M-Bus slaves, whereas an M-Bus master collects information from the meters and supports superior applications with it. The network topology is very flexible - star, ring and bus arrangements are supported. Besides the physical layer of the OSI model, M-Bus defines the data link, the network and the application layer. However, only the data representations defined by the application layer are relevant for this thesis. Therefore, an introduction into these parts of the application layer is given in the following.

Besides the `data send` command, there exists a number of ways a master can control and configure a slave (i.e., `application reset`, `set baudrate`, `synchronize action`). The way how meter data is represented in M-Bus is reflected by the slaves'

[1] http://www.m-bus.com

Fixed Data Header	Variable Data Blocks (Records)	MDH	Msg. specific data
12 Byte	variable number	1 Byte	variable number

Figure 3.2: M-Bus variable data structure in reply direction

responds to a `data send` command. There are two types of reply frames wherewith a slave can answer a master's request, either by a *Fixed Data Structure* or by a *Variable Data Structure* frame. Since data transported by a Fixed Data Structure frame can be seen as a simplified form of the Variable Data Structure, only the latter is described here. Besides that, the Fixed Data Structure frame type is not recommended for implementation anymore.

The Variable Data Structure APDU is shown in Figure 3.2. It consists of a Fixed Data Header, Variable Data Blocks (Records), and optional manufacturer specific data preceded by a Manufacturer Data Header (MDH). The Fixed Data Header, shown in Figure 3.3, contains static information about the metering device like an identification number, a manufacturer code, version information as well as status information informing about the device application. The Configuration field is intended for security extensions like the declaration of different modes of encryption and message signing.

Ident. Nr.	Manufr.	Version	Medium	Access No.	Status	Configuration
4 Byte	2 Byte	1 Byte	1 Byte	1 Byte	1 Byte	2 Byte

Figure 3.3: M-Bus fixed data header

The payload containing actual meter readings, their encoding, type and length is encapsulated in data records. Besides the current value, a data record can contain up to nine historical values which must be all of the same function. Possible functions are `Instantaneous value`, `Maximum value`, `Minimum value`, `Value during error state`. An arbitrary number of records forms the second field of the APDU shown in Figure 3.2. The structure of an M-Bus data record is illustrated in Figure 3.4. It consists of a Data Record Header (DRH) followed by the Data field. The header in turn is split up into a Data Information Block (DIB) and a Value Information Block (VIB). The DIB describes the function of the whole record, the length and encoding of the data field and gives information about the tariff assigned to each (historical) value. The engineering unit for each reading combined with a multiplier is encapsulated in the VIB.

DIF	DIFE	VIF	VIFE	Data
1 Byte	0-10 Byte	1 Byte	0-10 Byte	0-N Byte
DIB		VIB		
DRH				

Figure 3.4: Structure of an M-Bus data record

3.1.2 KNX

KNX [9] is a widespread standard in Home and Building Automation (HBA) systems. Developed and maintained by the KNX Association, it is also published as the ISO/IEC 14543-3-x family from 2007. It covers the majority of applications in HBA but finds a special use in room control, lighting and shading applications. The protocol defined by the KNX standard can be seen located at the field level of the automation hierarchy. The physical layer defines four different options of communication media and is characterised by narrow-bandwidth-communication. Available options are a twisted pair line (KNX TP), radio frequency communication (KNX RF), powerline communication (KNX PL) and also over IP-based networks (KNX IP) as a virtual medium. On top of the physical layer, the KNX standard defines a data link layer, network layer, transport layer and application layer.

The interworking model of KNX relies on a model of Functional Blocks (FBs). They give an abstract description of the behaviour of the devices. Correct interfacing between FBs is enabled by the definition of datapoint types (DPTs). In KNX, datapoints are also-called Group Objects (GOs) or Communication Objects (COs). COs have a Data Point Type (DPT) associated which provides the necessary semantic, i.e., metainformation to correctly interpret the meaning of a CO value. A DPT definition includes a datatype, the range of the CO value, the encoding and, if applicable, an engineering unit. Standard DPTs are defined for a wide range of Boolean, integer, and floating point values, as well as for a number of compound types. By the logical connection of COs of multiple devices being functionally compatible to each other (e.g., a dimmer and a light actuator) the desired distributed functionality of a KNX application is achieved. This logical connection is established by a group addressing scheme establishing a multicast communication group. Figure 3.5 shows a KNX interworking example between a light switch and a switching actuator. When the user pushes the button (1), the signal change (2) is processed by the user application (3) which updates the associated GO3 (4). This update is propagated to the KNX protocol stack (5) which looks up the associated group address 2/8/1 from the association table (6). This results in calling the GroupValue_Write service and the network stack sending an L_Data frame over the network (7). The protocol stack of the addressed device(s) recognises the destination address of the data frame in its association table (8) and looks up the associated GO (9). The change of state of the GO value is indicated to the user application of the switching actuator (10) which performs the desired action of driving the relay accordingly (12) and finally illuminating the lamp (13).

For the configuration of KNX devices, a number of different modes is available. The configuration phase encompasses the assignments of group addresses, i.e., the binding of devices, parametrisation of the user application and application program download. These features are variously supported by the different configuration modes. The System Mode (S-Mode) is the most comprehensive and flexible one and is based on the vendor-independent Engineering Tool Software (ETS) which is developed and published by the KNX Association. It also allows bus monitoring for diagnostics. Group address assigning can also be achieved by means of using the E-Mode (Easy-Mode). Depending on which

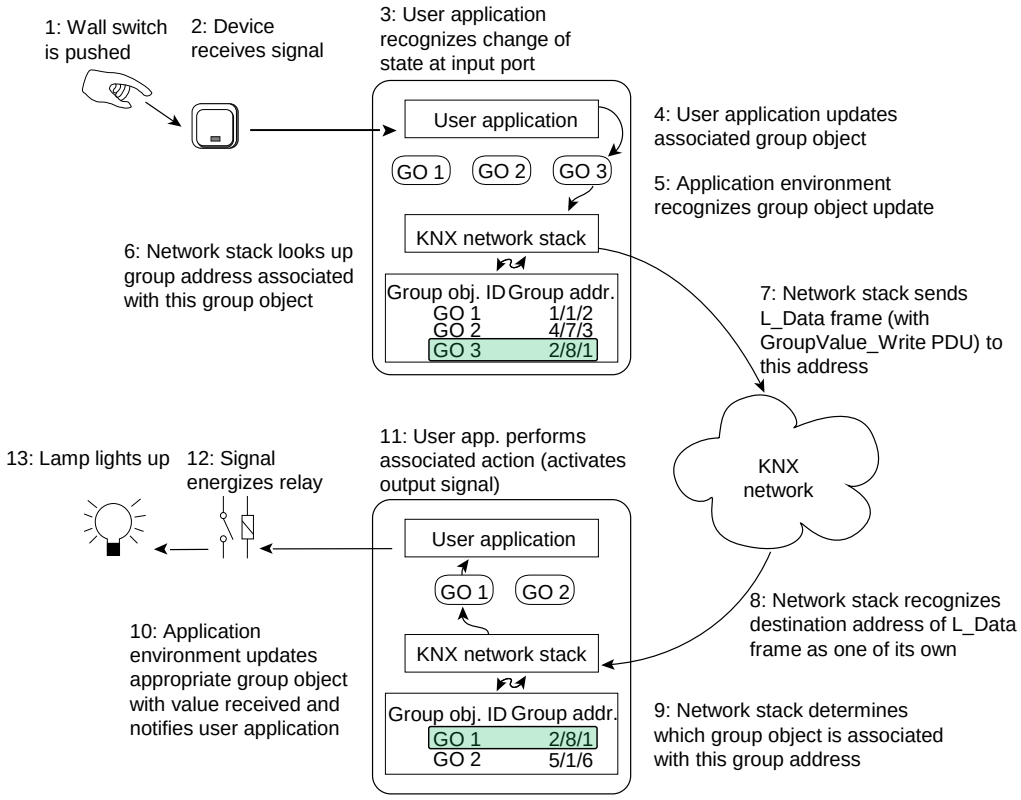

Figure 3.5: KNX interaction [138]

procedure the KNX device supports, this is an on-device configuration procedure via push buttons (PB-Mode), setting tag values on corresponding devices to identical values (LTE-mode) or using a special handheld device (Controller-Mode) to manage the binding.

3.1.3 Building Automation and Control Networks

The Building Automation and Control Network (BACnet) protocol was developed by the American Society of Heating, Refrigerating, and Air Conditioning Engineers (ASHRAE) and was standardized in 1995. Continuous maintenance and development are applied since then. Initially designed for the use at the management and automation level of the three tier automation hierarchy, nowadays BACnet has found a use in all kinds of building automation applications. The current standard is BACnet 2016 [56] which is also laid down as ISO 16484-5:2016 [11]. The network architecture encompasses a physical, a data link, a network, and an application layer of the ISO/OSI model whereas only the latter two layers are specified in the standard. To provide interoperability at the data link layer, five so-called network options describing the physical and the data link

layer are defined: Ethernet [139], Attached Resource Computer Network (ARCNET) [140], Master-Slave/Token Passing (MS/TP), Point-To-Point (PTP), and LonTalk [10]. Two normative annexes of the standard specify the transmission of BACnet messages over IP (BACnet/IP) and also by the use of ZigBee [141] as a wireless network option. The network layer provides two services for data transfer to the application layer, the `N-UNITDATA.request` and the `N-UNITDATA.indication` primitives. They represent an unacknowledged connectionless form of data transfer. The request is called by the application to initiate a data transfer, the indication informs the application about a reception of data.

With respect to the work presented, the application model is the most relevant part of the standard. As illustrated in Figure 3.6, it defines the *Application Process* as the part of the application which processes the information and handles the exchange of data between two peer applications. The application process in turn is divided into two parts: the *Application Program* and the *Application Entity* which is already part of the application layer. The former as well as the *Application Program Interface (API)* lying between the Application Program and the Application Entity are not specified by the standard. The *BACnet User Element*, which forms one half of the Application Entity, implements the service procedure portion of each application service. The other part is the *BACnet Application Service Entity (ASE)*. It represents a collection of five classes of services: *Alarm and Event*, *File Access*, *Object Access*, *Remote Device Management*, and *Virtual Terminal* which are responsible for information exchange between the application processes.

To allow remote devices to access process data, an object-oriented, network-visible representation of the stored data has been specified by BACnet. Up to now, 61 different *BACnet object types* are defined within the current BACnet standard. They differ in the composition of their so-called *BACnet properties* which can be seen as datapoints i.e., the logical representation of the process data of the technical process under control. Each property has a unique identifier referred to as `Property_Identifier`, a designated property type, and a conformance code attribute. The conformance code defines the access permissions of a property and specifies whether a property must be present or not. Possible values are Readable (R), Writable (W), and Optionally present (O). Figure 3.7 gives an example of such an object type definition. Vendors of BACnet devices are free to define their own proprietary object types (referred to as nonstandard object types) – even the definition of proprietary property types is possible. However, there are three mandatory properties that must be defined for each BACnet object: `Object_Identifier`, `Object_Name`, and `Object_Type`. The former two properties must be unique within a BACnet device. Since a BACnet object is always dedicated to exactly one device, these properties can be used to reference a BACnet object within the device.

Available BACnet object types as well as the included properties are mostly generic ones. For example, BACnet defines generic object types such as the `BACnet Binary Output Object Type` and the `BACnet Analog Input Object Type`. It is within the responsibility of the application program to map the functionality of a dedicated

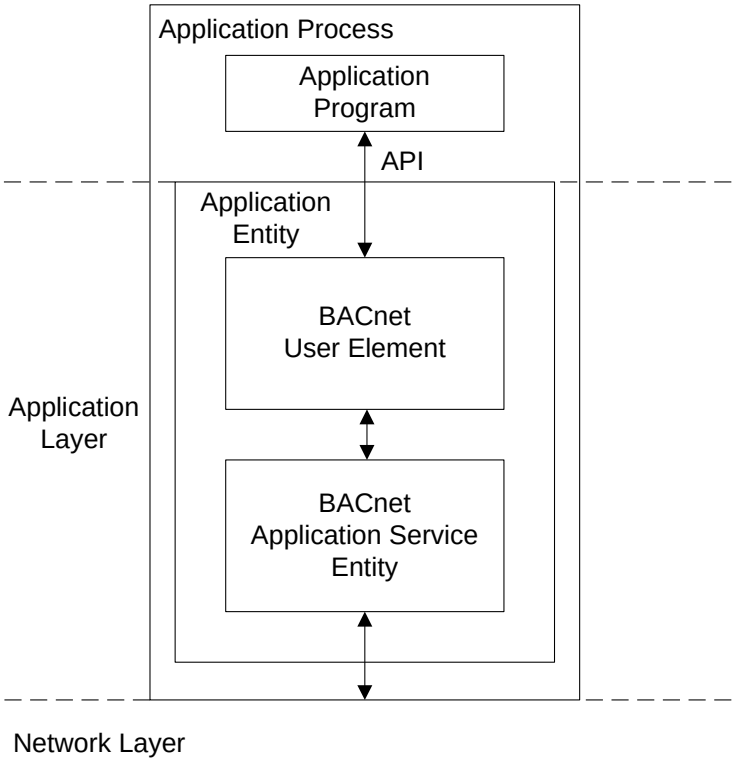

Figure 3.6: Model of a BACnet application process [56]

Property Identifier	Property Datatype	Conformance Code
Object_Identifier	BACnetObjectIdentifier	R
Object_Name	CharacterString	R
Object_Type	BACnetObjectType	R
Present_Value	REAL	W
Progress_Value	REAL	R
Resolution	REAL	O
Binary_Present_Value	BACnetBinaryPV	O
Output_Type	BACnetLightingOutputType	R
Lighting_Command	BACnetLightingCommand	W

Figure 3.7: BACnet object type definition [11]

application to certain BACnet objects. However, in the past few years, a considerable number of application specific objects have been defined, as for example to encapsulate basic lighting application properties. The definition of this BACnet `Lighting Output Object` is partly shown in Figure 3.7.

Every BACnet device holds exactly one special object called `Device Object`. The `Device Object` provides basic information about the BACnet device like vendor information, firmware and protocol version, and local time and date. Additionally, its

Object_Identifier and Object_Name must be unique within the whole BACnet network and so it can be used to identify the device in the network.

In addition to the representation of process data, the standard defines different communication services. Two important object access services are the ReadProperty and the WriteProperty services for getting and setting the value of a property. The ReadProperty service takes the Object_Identifier, the Property_Identifier and optionally the Property_Array_Index of the property that has to be read as arguments. If succeeded, the service response of a ReadProperty contains the input arguments of the request and the value of the property to be read. The WriteProperty service, on the other hand, takes the Object_Identifier, the Property_Identifier, the Property_Array_Index as well as the Property_Value and the corresponding Priority of the property that has to be written as arguments. Success is indicated to the client by sending a positive confirmation response.

In order to discover BACnet networks and to find specific devices, the unconfirmed Who-Is and Who-Has services are available. If one device broadcasts an unrestricted Who-Is to the network[2], every device (including the sender) responds with an I-Am service carrying the network address and the Object_Identifier of the respective Device_Object. By reading the Object_List of the Device_Object of every device that responded, all objects within the network can be determined. In case one BACnet device wants to discover the address information of the device holding an object from which the Object_Name or the Object_Identifier is known, it broadcasts a Who-Has service. The device that finds the requested object in its database returns an I-Have service. This response carries the address of the device together with the Device Object_Identifier as well as the Object_Identifier and the Object_Name of the requested object.

3.2 Integration of M-Bus smart meters

Since in the meanwhile modern buildings and production facilities are not imaginable without smart meters any more [142], [143], it is obvious to include resource consumption information delivered by these devices into management applications in the same manner than this is done for functional process data. The benefits arising thereof are improved monitoring and trending of these properties a technical process has. This facilitates the identification of optimisation potential, the detection of faulty conditions and is assumed to increase consumers' awareness on resource consumption [144].

To this aim, an OPC UA representation for the M-Bus application model is created. Hereby, a common standard for smart meter read out can be integrated into building management applications. An OPC UA server for M-Bus must be equipped with a

[2]The Who-Is can also be used to search for devices where the corresponding device ID is within a specific range. However, if the lower and upper bound of this range is set to the minimum and maximum Object_Id, all devices respond.

respective communication interface and a driver for application-level interaction with smart meters. It processes a client's access to its address space and propagates the requests to the target technology. Alternatively, the server can buffer an image of the meter data. This is especially recommendable if the target technology has limited resources, which is often the case for M-Bus devices. This way, the number of requests to these low bandwidth and often battery-powered components is reduced.

3.2.1 Mapping the M-Bus application model to OPC UA

In the following, an OPC UA representation of the M-Bus application model is presented. Therefore, a type model is defined in a first step. Figure 3.8 shows the type hierarchy for the definition of an OPC UA information model for M-Bus which is based on modelling concepts defined in the OPC UA for Devices (DI) specification [145]. The information model defined in this companion specification facilitates finding OPC UA representations of automation technologies with a device-centric view. It defines a `DeviceType` ObjectType (a subtype of the `BaseObjectType` and the `TopologyElementType`) with a predefined set of properties (including a `SerialNumber`, a `Manufacturer` property and revision information) each device model instance may encompass. These property instance declarations all have the so-called ModelingRule `Optional`. This means, properties with this ModelingRule may optionally be present for each instance of this ObjectType. As a subtype of the `DeviceType`, a new OPC UA ObjectType is defined, the `M-BusMeteringDeviceType`. It inherits all the properties of its supertype (which are only shown in Figure 3.8 for the `DeviceType`) and also defines additional `Medium` and `Status` properties. This set of properties is dedicated to reflect static information about an M-Bus metering device. The mapping of information from the Fixed Data Header of the M-Bus APDU to the `M-BusMeteringDeviceType` is defined in Figure 3.9. However, some of the properties of the OPC UA model do not have an M-Bus equivalent. So they can be omitted. This is in compliance with the DI specification.

The actual meter readings of the M-Bus device which are the content of Variable Data Blocks (i.e., the records) are mapped to another OPC UA ObjectType, the `M-BusDataRecordType` as illustrated in Figure 3.8. The `M-BusDataRecordType` has a property `Function` and a variable `Reading`. The `Function` property reflects the Function Field of the DIB in a Variable Data Block.

The `Reading` variable InstanceDeclaration is of the `AnalogItemType` which is taken from the *Data Access* [146] part of the OPC UA specification. Its `EngineeringUnit` property which is part of the definition of this VariableType represents the Unit field of the M-Bus VIB. `EURange` is another mandatory property of the `AnalogItemType`. It defines the value range of this variable under normal operation. In this model, `EURange` reflects the length and encoding given by the M-Bus DIB. In order to enrich this model with tariff information, the `Reading` variable is extended by an additional `Tariff` property. The Tariff field of the M-Bus DIB is mapped to this property. Since an M-Bus compliant meter can provide access to up to nine historical values encapsulated in one record, concepts from the OPC UA Historical Access specification (OPC UA Part 11)

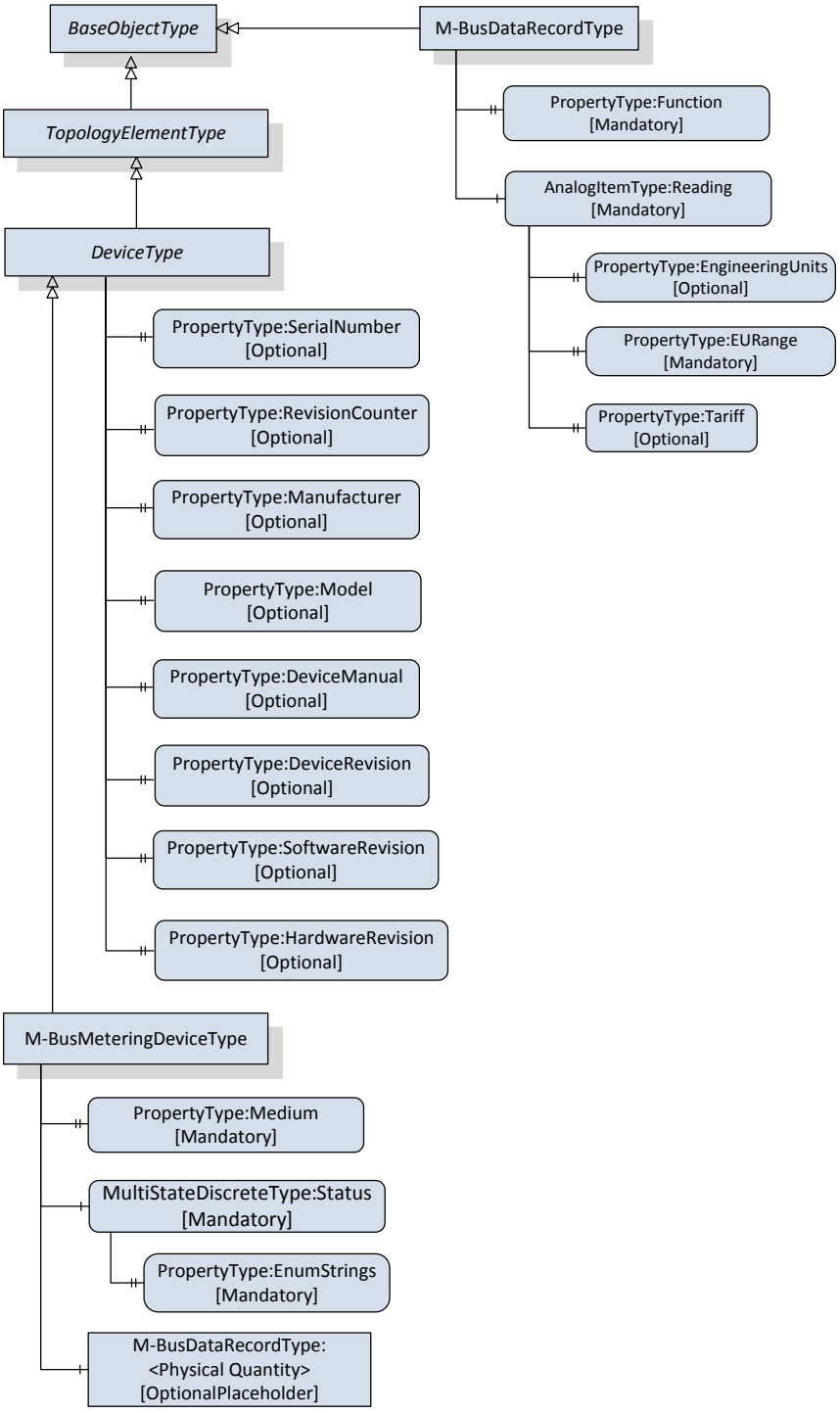

Figure 3.8: OPC UA ObjectType definitions for M-Bus

M-Bus APDU field	OPC UA Property
Identification Number	SerialNumber
Manufacturer	Manufacturer
Version	DeviceRevision
Medium	Medium
Status	Status

Figure 3.9: Mapping of M-Bus fixed data header

could be included in this information model. However, this is out of focus of the current version of the M-Bus representation.

Returning to the `M-BusMeteringDeviceType` definition in Figure 3.8, it can be seen that an instance of this type can have arbitrary many instances of the `M-BusDataRecord` type. This is achieved by using the `OptionalPlaceholder` ModellingRule. Hereby, the fact is reflected that an M-Bus device can provide multiple readings like the daily and the cumulated consumption of the media under measurement or even other medium-specific physical quantities.

3.2.2 Instantiating the M-Bus/OPC UA object types

Figure 3.10 shows an exemplary instance of the `M-BusMeteringDeviceType`. It represents a real-world *Siemens WFM21* M-Bus heat meter including the data it provides. Corresponding to the type definition in Figure 3.8, an M-Bus metering devices has a number of static attributes (i.e., properties) and naturally delivers measurement values including status information. Contrary to the type definition, the instance in Figure 3.10 is provided with concrete values. The WFM21 delivers, besides others, the measurement values of the *cumulated energy* and the *cumulated volume* of heating water. Therefore, the `HeatMeter` object references two objects representing M-Bus data records, the `CumulatedEnergy` object and the `CumulatedVolume` object. The `Function` properties of both records are set to `CurrentValue`. The heat meter in this example is also capable of delivering minimum, maximum and average values which is disregarded here, however. The value attribute of the `Reading` variables is set to the actual readings calculated from the M-Bus Data and the Multiplier field of the VIB. Accordingly, the `EngineeringUnits` and `EURange` property values are set, too. The `CumulatedEnergy` record of the WFM21 also delivers tariff information. This is regarded in this model instance by the optionally instantiable `Tariff` property, whose value attribute is set to `Tariff1`.

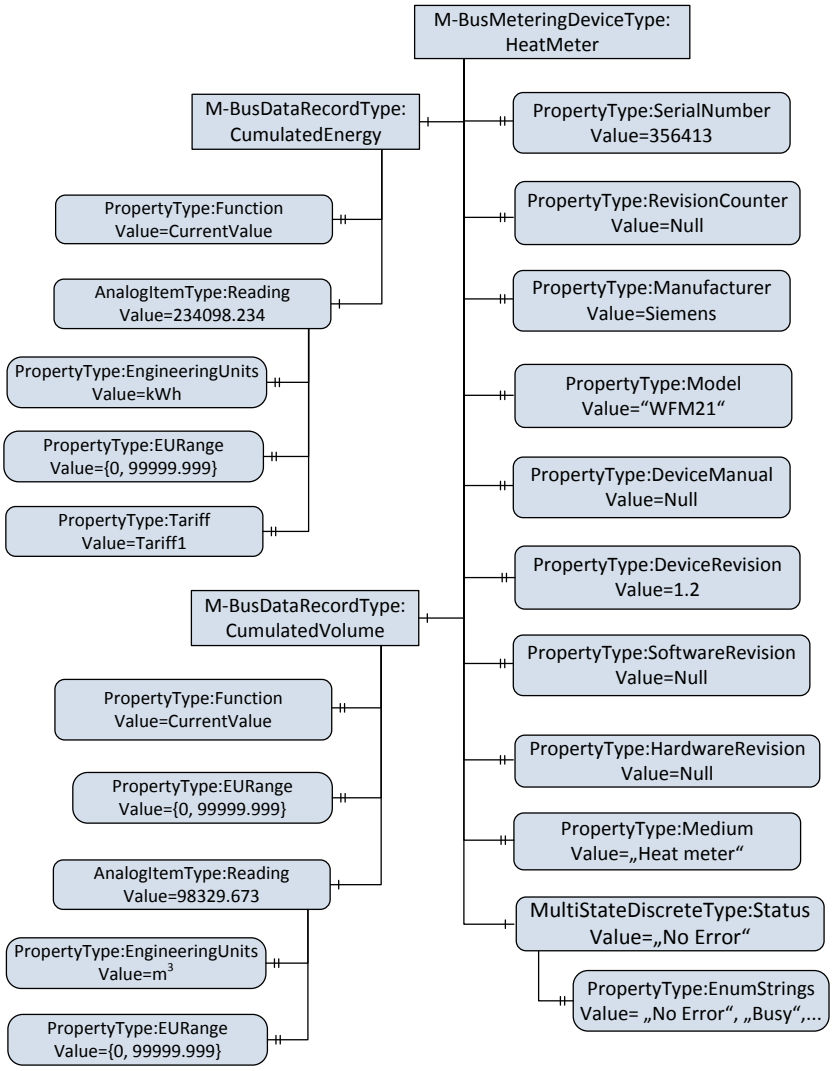

Figure 3.10: Instantiation of an exemplary M-Bus representation

3.3 Integration of KNX

By now, first steps towards OPC UA information models for KNX have already been made [147]. This preliminary work will be enhanced by integrating meta information into the existing model. Since an information model only gives a description about the general scheme how KNX process data (i.e., KNX function blocks, group addresses) are represented and accessible, it cannot state anything about the actual configuration of a distinct installation. Therefore, the types defined in a KNX/OPC UA information model need to be instantiated in a way such that they map the actual configuration of the target KNX network. Therefore, the XML export functionality of the ETS4 is used to transfer the configuration data of a distinct KNX installation to an OPC UA server.

Since the interworking model of KNX is based on FBs, a way had to be found to map these KNX FBs to OPC UA. FBs are defined by the KNX specification and give an abstract description of the behaviour of KNX devices. They consist of different datapoints for input data, output data and parameters. Each datapoint in KNX has a well-defined datapoint type (DPT) to ensure that data can be exchanged between different FBs in an unambiguous way. In order to achieve the requested functionality, a KNX device implements distinct FBs. To reach the desired distributed control functionality, e.g., by binding a KNX dimmer to a KNX dimming actuator, datapoints belonging to the respective FBs are logically linked by assigning them so-called *group addresses*.

The preliminary work about the information model for KNX mainly addresses the plain data representation of the standard. It describes a schema how KNX data types can be defined by OPC UA structured data types. The datapoints of KNX FBs are mapped to user defined OPC UA variable types which get the regarding data types assigned to their `Datatype` attribute. KNX FBs are mapped to OPC UA complex object types which reference the variables defined before, in order to finally create a representation of KNX FBs in OPC UA.

Also the different ways of addressing datapoints in KNX are covered by this model. Therefore, various kinds of variable types are defined which are used to carry the different types of addresses used in KNX. These address variables are assigned to the particular datapoints by the user-defined reference type `HasKNXAddress`. Figure 3.11 shows an example of three FBs describing a dimming application. The OPC UA complex object types are instantiated to complex objects to represent live data in a KNX installation. In this example, two dimmers are logically linked to a dimming actuator. For clarity, this is only shown for the `SwitchOnOff` datapoints which get the `DimmingGroup1` group address of the value "`1/1/0`" assigned. Further it can be seen that the `OnOffAction` parameter of `Dimmer2` FB is addressed by the individual address of the device, an object ID and a property ID. This is modelled by the `Dimmer2Object5Property6` variable.

3.3.1 ETS4 project structure

In previous versions of the ETS, different views on a KNX installation project were already available. This allowed the user to organise the devices and datapoints of a KNX

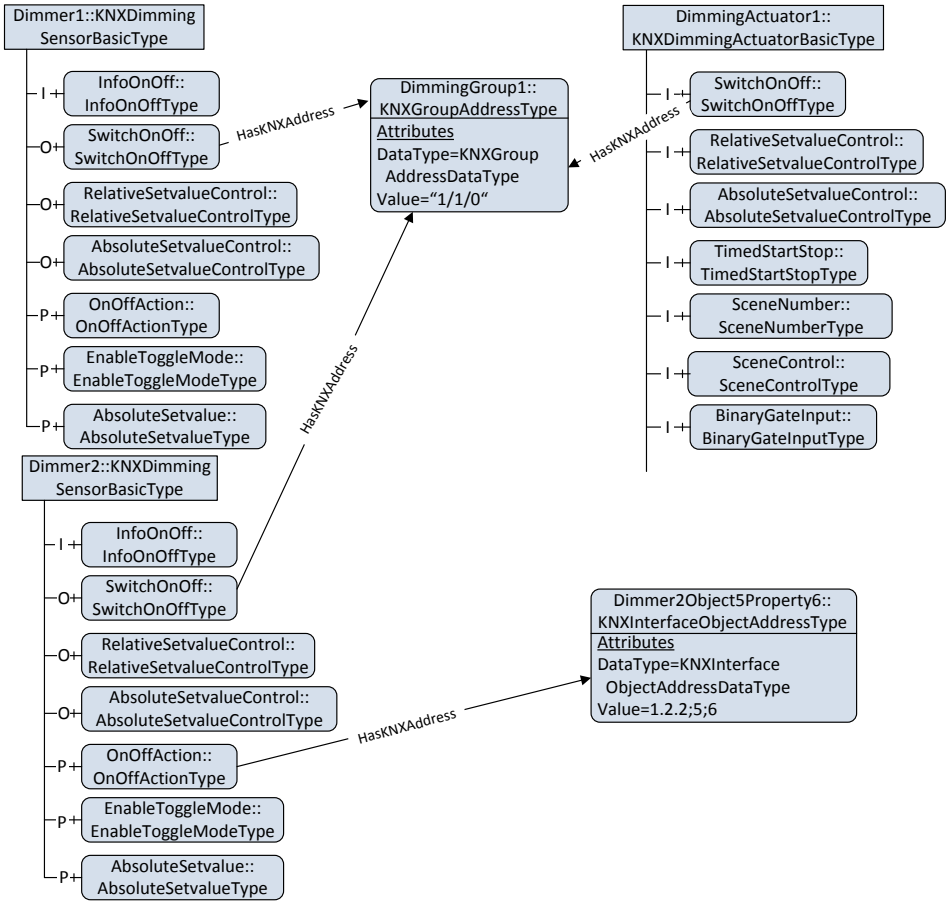

Figure 3.11: KNX functional blocks and addressing datapoints in OPC UA [147]

installation by different criteria. For device configuration absolutely necessary views are the *Device view* and the *Group Address* view where the parametrisation of KNX devices takes place and group addresses are assigned. The *Topology view* which can also be considered relevant for device configuration shows a hierarchical structure of devices organised by their individual addresses. This reflects the network topology scheme of a KNX network consisting of *Areas*, *Lines* and *Device* addresses. Another view which was already available in the ETS3 is the *Buildings view*. Here, the physical structure of buildings is represented by a hierarchy of sub-elements like floors, corridors and rooms which in turn can have KNX devices associated. This allows an assignment of network components to different locations of a building.

A feature which is new to the ETS4 is the *Trades view*. This view emphasises the functional affiliation of KNX devices. It allows to distinguish between different trades like lighting, Heating, Ventilation and Air Conditioning (HVAC) or safety applications

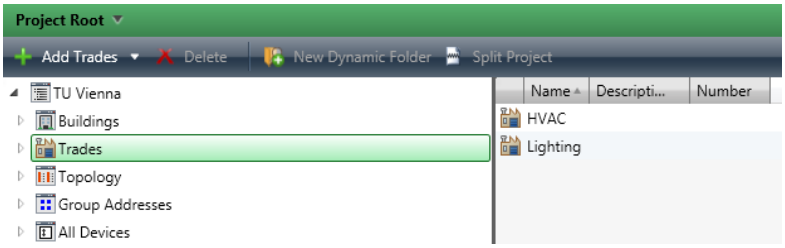

Figure 3.12: Project structure in the ETS4

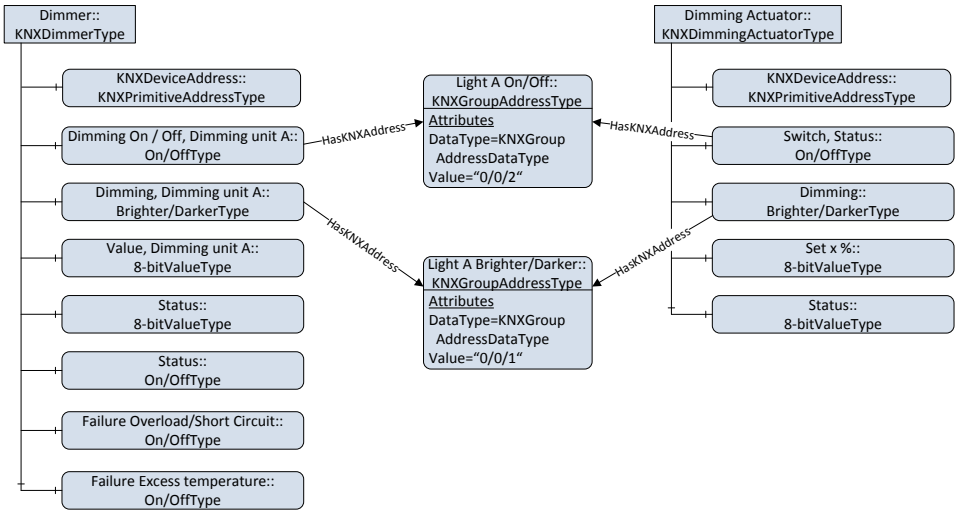

Figure 3.13: Adapted OPC UA model for KNX

and lets these trade and sub-trade nodes expose devices whose functionally refer to them. Figure 3.12 shows the structure of a project in ETS4 with the different views described before. In the Trades view, for example, two representatives are visible: HVAC and Lighting.

3.3.2 Mapping the ETS4 project structure to OPC UA

Vendors of KNX devices implementing the application programs of their products need to follow the interworking model of KNX which is based on FBs. However, the ETS does not use this concept of KNX FBs. Instead, it uses a different model where the KNX device models available in the ETS product catalogue include a set of so-called *COs* specific for each device. These COs can be seen as a representation of the datapoints of FBs the respective devices implement. Like datapoints in KNX the COs have a distinct datatype to assure type safety. Also binding of two or more COs to each other is done by the assignment of a group address.

In order to be able to use the XML export data of the ETS4 to configure an OPC UA server, the previous KNX information model needs to be adapted. Figure 3.13 exemplarily shows the instances of two OPC UA object types standing for dimmer and dimming actuator devices. Using `HasComponent` references, they now expose the COs defined by the device model from the ETS catalog. The model of a universal dimmer device for example includes, among others, a `Dimming On / Off`, a `Dimming` and a `Value` CO. In this adapted information model, OPC UA variables are used to represent these COs (like in the previous model to map the datapoints of FBs). When defining the corresponding variable types, the attributes specific for this OPC UA node class need to be set. The `BrowseName`, the `DisplayName` and the `DataType` attributes, to name a few, are defined following the same method like presented in the precedent work [147]. Figure 3.13 shows a part of the group address binding necessary for a dimming application. The respective COs (e.g. the `Dimming, Dimming unit A` CO of the dimmer and `Dimming` CO of the dimming actuator) are both linked to a group address variable using a `HasKNXAddress` reference and hereby bound to the group address `0/0/1`.

The meta information included into an ETS4 KNX installation project is not only useful at configuration time but also for management applications supervising a live KNX installation. For example, it can be integrated into a Building Management System (BMS) and hereby support the operator by providing additional information about the BAS under control. One key feature of OPC UA is enhancing process data by its semantics. This shall be utilised here such that also the meta information provided by the ETS4 is mapped to OPC UA.

In the following, a way is introduced how this mapping is performed for each of the views in the ETS4 containing valuable semantics for a management application in a BAS. The focus is set on the Buildings view, the Trades view, the Topology view and the Group Address view.

One benefit which arises from the integration of meta information from the single ETS4 views into a holistic KNX model is that every time the KNX configuration is changed via the ETS4 this information is available via XML export and therefore can be used to keep a management level application using this model synchronised (cf. Section 3.4.3). The ETS4 is the central configuration tool for KNX devices and consequently it is easy to keep the actual configuration of the physical devices and the configuration of an ETS4 project aligned. The only constraint to retain configuration data consistency is that no changes may be made using the E-Mode.

Group Addresses view

In this view, the organisation of main groups, middle groups and sub groups defined in the ETS4 project is shown. Group addresses in KNX are triples composed of the addresses of

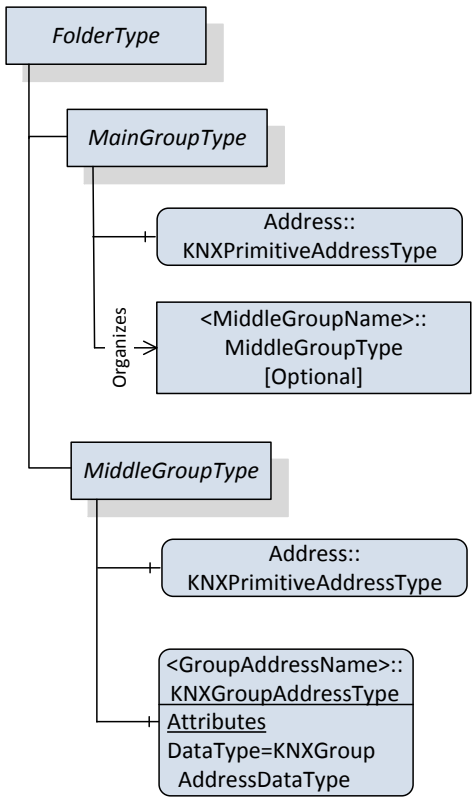

Figure 3.14: ETS4 group address view in OPC UA - object type definition

the hierarchical groups in the form `main group/middle group/sub group`[3]. The mapping to OPC UA works similar like in the topology view. Complex object types (shown in Figure 3.14), the `MainGroupType` and the `MiddleGroupType`, each referencing a variable InstanceDeclaration are defined. These `KNXPrimitiveAddressType` variables are used to carry the address component of the respective group. The definition of the `MainGroupType` also consists of a `MiddleGroupType` InstanceDeclaration whereas the `MiddleGroupType` exposes an InstanceDeclaration of a `KNXGroupAddressType` variable [147]. This variable does not only get the address value of the sub group assigned but the complete group address triple. The reason for that is that the group address value needed for addressing COs can hereby be read by an OPC UA client in one step. Otherwise it needed to browse to the other two address variables as well to gain the whole address information. This would significantly increase the response time of a management client. An instance example of this group address model is depicted in Figure 3.15.

[3]In KNX there also exists a two-level addressing scheme which is not considered in this work. However, the mapping to this information model works analogously

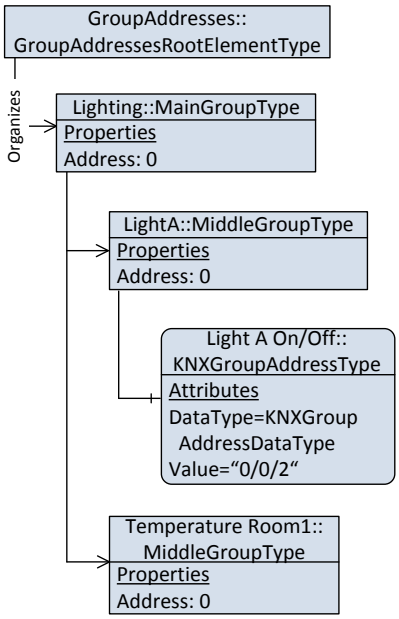

Figure 3.15: ETS4 group address view in OPC UA - example of instances

Topology view

The location of a KNX device in the Topology view reflects its individual address, which is a triple consisting of an `Area` field, a `Line` field and a `DeviceID`. So, each single component of an individual address carries one part of the address information needed to uniquely identify a KNX device in the network. In the OPC UA information model (cf. Figure 3.16), this is taken into account by defining complex object types for each component. The `KNXAreaType` as well as the `KNXLineType` reference an `Address` variable of the type `KNXPrimitiveAddressType` [147]. To express the fact that addresses can have changeable values, the OPC UA built-in `HasComponent` reference types are used to link the variables to the object types. `HasComponent` reference types in OPC UA are contrary to the built-in `HasProperty` reference types used for referencing data variables. For a deeper understanding of this concept, confer to [32] Part 3.

In order to model the hierarchy of the three address components of a KNX individual address, the definition of the `KNXAreaType` object type references a `KNXLineType` InstanceDecalaration whereas the `KNXLineType` object type definition references a `KNXDeviceType` InstanceDeclaration. As before, `Organizes` reference types are taken to link these nodes. An example how this part of the information model can be instantiated is shown in Figure 3.17. It illustrates an area (`Area1`) exposing two lines, `Line1` and `Line2`, where each line contains one KNX device. Hence, the `Dimmer` device has the individual address `1.1.1` and the `DimmingActuator` is addressed by `1.1.2`.

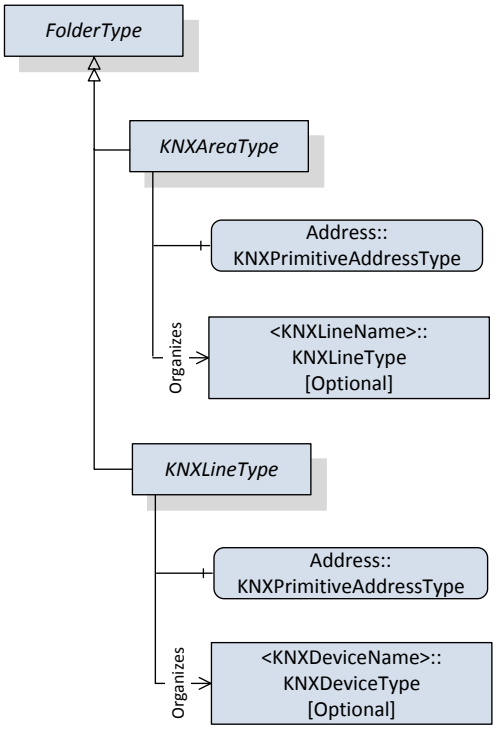

Figure 3.16: ETS4 topology view in OPC UA - object type definition

Buildings view

Modern BMSs provide a live process image of datapoints of KNX installations as well as a view on data originating from other network technologies. Having the physical location of a KNX device and its corresponding datapoints available in this live view has several benefits. It facilitates the visualisation of datapoints in a graphical model of a building and can help to improve the usability of a BMS. Especially, it enables the assignment of alarms or events triggered somewhere in the network to a distinct location in a building in a very easy way. Consider, for example, an alarm indicating a lamp has failed. By following the references from the datapoint which triggered the alarm to the building part entity a precise order can be issued to the facility manager to replace the lamp.

Figure 3.18 shows the definition of OPC UA object types representing the hierarchy of entities in a building. A building in the Buildings view of the ETS4 can consist of building parts, stairways, floors, rooms, corridors and cabinets. In the model in Figure 3.18, this is reflected by the definition of a `BuildingType` object type which is a subtype of the built-in `FolderType`. *InstanceDeclarations*[4] of object types referring to these parts of

[4]When instantiating an ObjectType node in OPC UA which exposes InstanceDeclarations the latter are also instantiated, depending on the ModellingRule

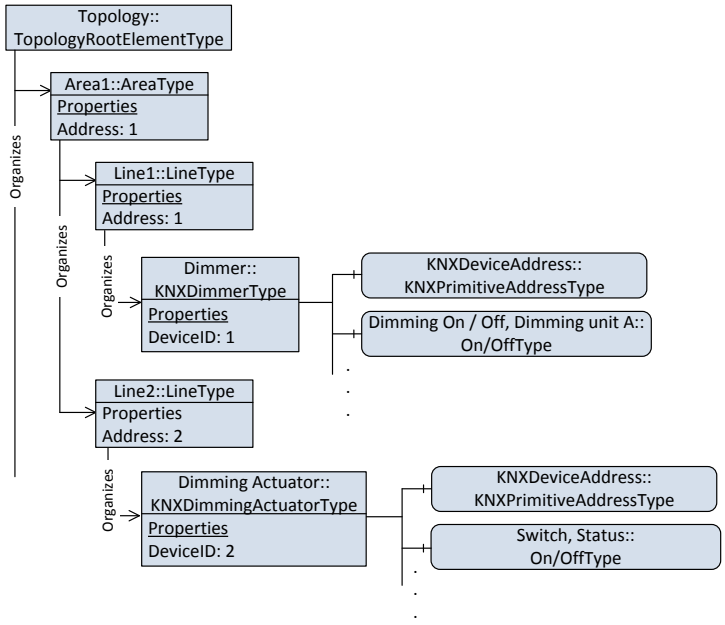

Figure 3.17: ETS4 topology view in OPC UA - example of instances

a building are linked to the `BuildingType` using `Organizes` reference types. In the context of folders in OPC UA whose main purpose is to organise the address space it is common to use the `Organizes` reference type. The `[Optional]` labels within the InstanceDeclarations denote *ModellingRules*. They reflect the fact that all the building part entities can optionally be present.

The `BuildingPartType` is modeled as a subtype of the `BuildingType` which is indicated by the `HasSubtype` reference type. Thus, all the InstanceDeclarations referenced by the `BuildingType` object type are inherited to its subtype. This follows from the definition of the OPC UA address space and is not shown in this illustration for reasons of clarity. The `FloorType`, following the schema of the ETS4 project structure, references a `RoomType`, a `CorridorType` and a `CabinetType` InstanceDeclaration. Reaching the lowest level of this hierarchy of building part entities, the definitions of the `StairwayType`, the `RoomType`, the `CorridorType` and the `CabinetType` each reference a `KNXDeviceType` InstanceDeclaration. Like in an ETS4 project, these are the locations where KNX devices are allowed to be present.

Figure 3.19 shows a practical example where the types defined before are instantiated. It reflects a site consisting of two buildings, `BuildingA` and `BuildingB`. The former exposes a `Floor4` which in turn has a `Room1` where a dimmable lighting is installed. Therefore, a KNX `Dimmer` and a `DimmingActuator` are placed. The COs implemented by the devices are shown on the right side of this figure.

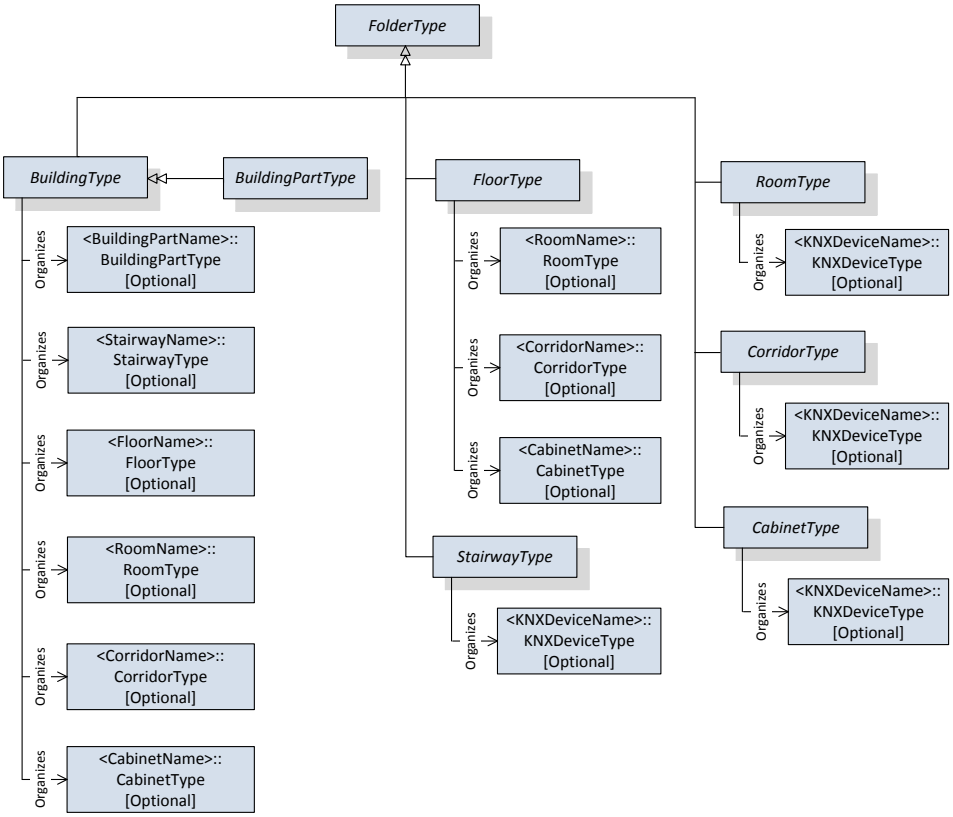

Figure 3.18: ETS4 buildings view in OPC UA - object type definition

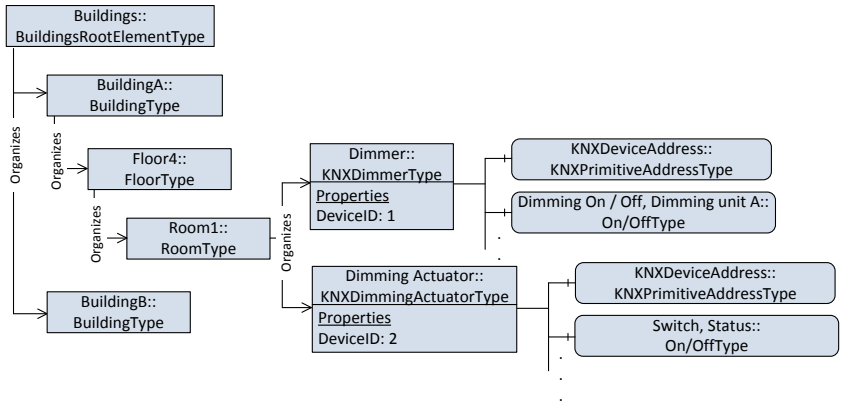

Figure 3.19: ETS4 buildings view in OPC UA - example of instances

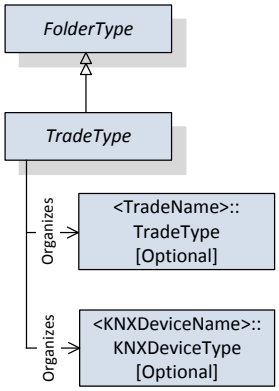

Figure 3.20: ETS4 trades view in OPC UA - object type definition

Trades view

A conceivable use case for a building management application implementing this trade model is the trade-based aggregation of energy consumption data. It is straightforward to filter data belonging to a distinct trade by traversing the references from the trade of interest down to the associated KNX devices. Subsequently, the datapoints representing the energy consumption data are read from the selected devices.

Including the schema from the Trades view of the ETS4 into the proposed information model is done similarly like for the Buildings view. An object type definition also being a subtype of the OPC UA `FolderType` is introduced like shown in Figure 3.20: the `TradeType` object type. By using `Organizes` reference types, a `TradeType` InstanceDeclaration and a `KNXDeviceType` InstanceDeclaration are linked to the superior type. This recursive definition of the `TradeType` results from the fact that a trade in the ETS4 can expose a hierarchy of sub trades of an arbitrary depth.

Figure 3.21 exemplarily shows how this model can be instantiated. The `Lighting` trade consists of two devices, a dimmer and a dimming actuator. The `HVAC` trade in turn exposes a `RoomTemperatureController`.

3.3.3 Instantiating KNX/OPC UA object types

Compared to other building automation standards, KNX has the advantage that in S-Mode all the necessary configuration is done with the standard engineering tool ETS. Thus, the whole configuration data of KNX projects is available in ETS. Since the release of ETS4, the whole project configuration can even be exported as XML files. These files give a complete description of the whole KNX installation parameterised in an ETS4 project. This includes not only the configuration of the KNX devices and group addresses but also meta information of the KNX installation like the network topology and the building structure.

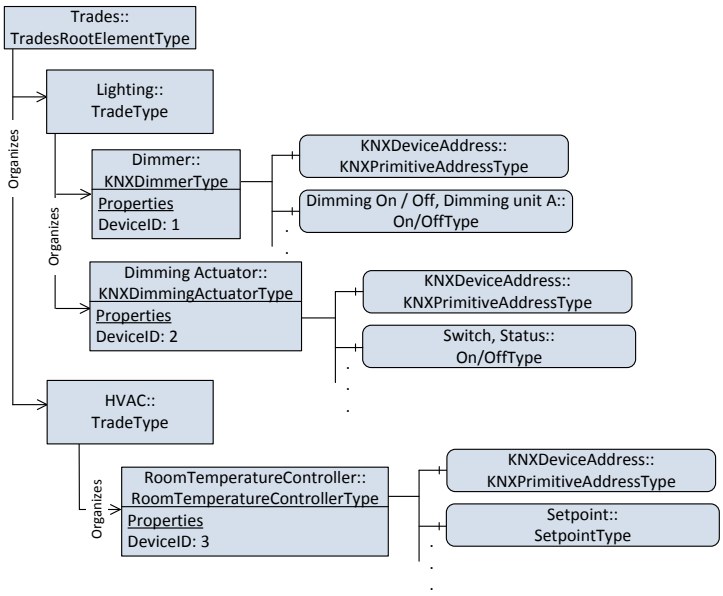

Figure 3.21: ETS4 trades view in OPC UA - example of instances

The XML export functionality of ETS4 shall be used to provide an easy mechanism to configure KNX/OPC UA servers in a fully automated way. In other words, the KNX/OPC UA servers import the XML description exported by ETS4 which is then used to instantiate the corresponding OPC UA types defined by the KNX information model. Using this mechanism, the full ETS4 project structure is available in OPC UA and thus available by OPC UA clients.

In order to clarify the transfer of ETS4 data to OPC UA, a simple ETS4 project has been created which will be used as an example in the following. A screenshot of this project is shown in Figure 3.22. The following XML listings as well as the final model in Figure 3.23 correspond to this ETS4 project.

The XML file named "0.xml" which is generated when exporting an ETS4 project is basically organised in four different elements, which are identified the by associated tags: `<Topology>`, `<Buildings>`, `<GroupAddresses>` and `<Trades>` which in turn consist of sub-elements corresponding to the entities present in a distinct view. These elements contain attributes specific for the respective ETS4 view which will be mapped to the value attributes of the OPC UA objects and variables to be instantiated thereof. In the following, the mapping will be described for every XML element representing a view. Excerpts of the relevant parts of these elements are listed, too.

A mapping strategy which applies to all the elements is that the value of the NodeId of the resulting OPC UA object is set to the ID value of the respective element. The ID values in the XML file are unique and hereby an unambiguous mapping to OPC

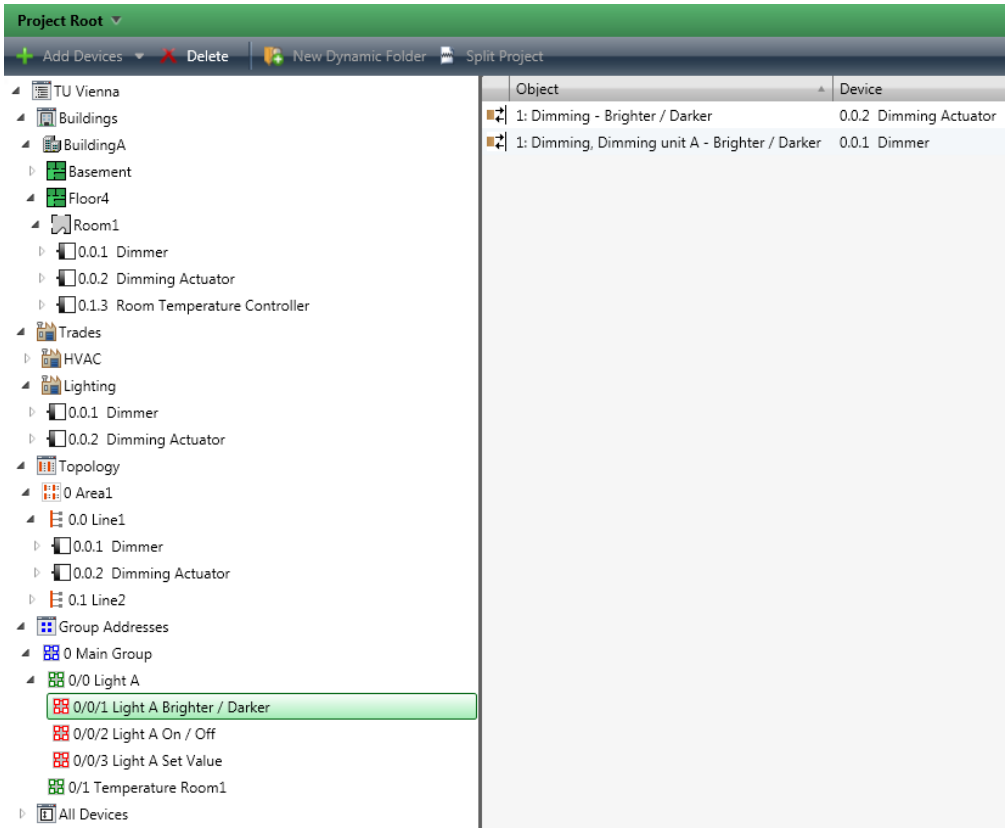

Figure 3.22: ETS4 project

UA nodes is achieved. The way how the references need to be set can be gained from the hierarchical order in which the XML elements are organised. This order results in `Organizes` reference types pointing from the super element to the sub element.

The `<Topoplogy>` element like seen in the listing below exposes a hierarchy of `Areas`, `Lines` and `DevicesInstances` represented by an `Id`, an `Address`, a `Name` and several other attributes which are not considered in this model. The `Name` attributes are assigned to the `BrowseName` and to the `DisplayName` attributes of the instances of `AreaType`, `LineType` and `DeviceType` OPC UA object types. The `Address` attributes of the XML elements are transferred to the value attributes of the `Address` variables referenced by the latter object types introduced in Section 3.3.2. This can be seen in the topology branch of the model in Figure 3.23. In the XML data a `DeviceInstance` organises a set of references to COs in the form of elements labeled `ComObjectInstanceRef`. By a `RefId` attribute (for the first `ComObjectInstanceRef` element it has the value `M-0001_A-9067-03-202B_O-0_R-205`) the logical connection to the description of the respective CO is made. The description of each CO is located in separate XML files. In here, all the parameters related to a CO are laid down, for example the name, the priority and the flag values. Having these

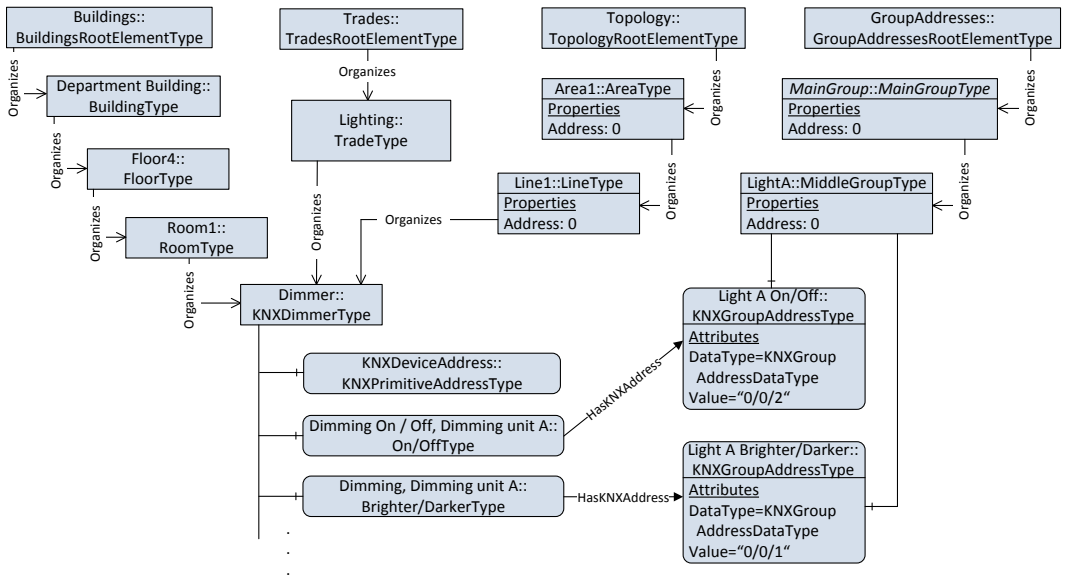

Figure 3.23: KNX dimmer device and its associated physical location, trade, topological location and group addresses

data available, an assignment of the `ComObjectInstanceRef` to the OPC UA variable representing the CO can be made. In this example, the `ComObjectInstanceRef` with the ID `M-0001_A-9067-03-202B_O-0_R-205` corresponds to the `Dimming On / Off, Dimming unit A` CO of the dimmer device in Figure 3.23.

Each `ComObjectInstanceRef` exposes a `Connectors` element which organises the references to the associated group addresses. The group addresses are linked to the CO using the `GroupAddressRefId` attribute.

```
<Topology>
 <Area Id="P-032A-0_A-0" Name="Area1" Address="0" CompletionStatus=
       "Undefined">
  <Line Id="P-032A-0_L-0" Name="Line1" Address="0" ...>
   <DeviceInstance Id="P-032A-0_DI-1" Name="Dimmer" ...
                Address="1" ...>
    <ComObjectInstanceRefs>
     <ComObjectInstanceRef RefId=
        "M-0001_A-9067-03-202B_O-0_R-205" IsActive="1">
      <Connectors>
       <Send GroupAddressRefId="P-032A-0_GA-2" />
      </Connectors>
     </ComObjectInstanceRef>
                    .
                    .
                    .
```

The value (`P-032A-0_GA-2`) of this `GroupAddressRefId` appears again at the `Id` attribute of the lower `GroupAddress` element in the following listing. Here, the section

of the XML data dedicated to the Group Addresses view is shown. It is structured into two nested `GroupRange` elements where the inner one contains the actual `GroupAddress` elements. This hierarchy matches the segmentation of KNX group addresses in main groups, middle groups and sub groups. The outer `GroupRange` element stands for a main group, the inner one represents a middle group and the `GroupAddress` elements map the sub group. Like proposed for the previous view elements, the `Name` attributes of the XML elements are transferred to the `BrowseName` and to the `DisplayName` attributes of the OPC UA object instances derived thereof. The `Address` field of the `GroupAddress` elements is mapped to the sub group field of the value attribute of the `KNXGroupAddress` OPC UA variables of this model. The value of main group and the middle group component follows from the location of the `GroupAddress` element in the hierarchical XML description. The result of this mapping can be seen in the right branch in Figure 3.23.

```
<GroupAddresses>
 <GroupRanges>
  <GroupRange Id="P-032A-0_GR-1" Name="Main␣Group" RangeStart="1"
  RangeEnd="2047">
   <GroupRange Id="P-032A-0_GR-2" Name="Light␣A" RangeStart="1"
               RangeEnd="255">
    <GroupAddress Id="P-032A-0_GA-1" Address="1" Name=
        "Light␣A␣Brighter␣/␣Darker" />
     <GroupAddress Id="P-032A-0_GA-2" Address="2" Name=
        "Light␣A␣On␣/␣Off" />
            .
            .
            .
```

In the `Buildings` element of the XML export file the content of the ETS4 Buildings view is represented. The levels of the `BuildingPart` elements correspond to the levels where the entities are located in the ETS4 project. Each XML element contains of an `Id`, a `Name` and a `Type` attribute. The `Type` attribute declares the type of the element, for example `Building` or `Floor` like seen in the listing below. The bottom level XML element, the `DeviceInstanceRef` is used to reference the devices located in a distinct building part entity. In the `Topology` element, where the devices are defined, each device gets a `DeviceInstance Id` assigned. Its value is used to identify a device. In this example, the dimmer device with the ID `P-032A-0_DI-1` (indicated by the `RefId` field) is located in `Room1` of `Floor4` in `BuildingA`, like illustrated in Figure 3.23.

```
<Buildings>
 <BuildingPart Id="P-032A-0_BP-0" Name="BuildingA"
        Type="Building" ...>
  <BuildingPart Id="P-032A-0_BP-1" Name="Floor4" Type="Floor" ...>
   <BuildingPart Id="P-032A-0_BP-2" Name="Room1" Type="Room" ...>
    <DeviceInstanceRef RefId="P-032A-0_DI-1" />
     <DeviceInstanceRef RefId="P-032A-0_DI-5" />
            .
            .
            .
```

The XML view element which is still remaining is the `Trades` element like listed below. The attribute of the `Trade` element relevant for this information model is the `Name`. It is again mapped to the `BrowseName` and to the `DisplayName` of the resulting OPC UA objects, which can be seen at the trades branch in Figure 3.23. The `DeviceInstanceRef` elements declare the KNX devices assigned to a distinct trade. The dimmer device (`RefId="P-032A-0_DI-1"`) mentioned before is part of the `Lighting` trade which is also reflected in the OPC UA model.

```
<Trades>
 <Trade Id="P-032A-0_T-0" Name="Lighting"
        CompletionStatus="Undefined">
  <DeviceInstanceRef RefId="P-032A-0_DI-1" />
  <DeviceInstanceRef RefId="P-032A-0_DI-2" />
 </Trade>
<Trades>
```

3.3.4 Proof of concept

To show the practical feasibility an existing commercial building management server implementation, the so-called Voyager BMS Server developed by the company NETx-Automation[5] has been prototypically extended. Among other features, this product implements an OPC UA server and connectivity to KNXnet/IP interfaces. Hereby it allows to integrate KNX installations by means of OPC UA. Within the research project "iModelA", an importer module for ETS4 project data has been developed. It takes a ".knproj" export file containing the information about a KNX ETS4 project. The file is extracted, its XML content is processed. Following the mapping scheme of KNX project information to an OPC UA address space like introduced before, the respective instances are created in the OPC UA servers address space. The resulting address space exactly corresponds to the information of the originating ETS4 project. This allows an OPC UA client exploring the server to have the same view on the KNX installation as defined in the ETS4 project - naturally including the runtime values of the KNX datapoints.

3.4 Integration of BACnet

In order to realise the integration of BACnet, OPC UA servers need to be included into the hierarchical communication infrastructure of building automation networks. They are used to collect data from different BACnet controllers to expose a live process image to e.g., an overlying BMS. The latter encompasses an OPC UA client which connects to this OPC UA server. The client can be either in the same network or even at a remote location. In this case, the firewall of the router must be configured properly such that it passes the connection requests of the OPC UA client to the network.

[5] https://www.netxautomation.com/netx/en/products/server/bms-server

3.4.1 Mapping the BACnet application model to OPC UA

In order to enable OPC UA clients to access a building automation network via an OPC UA server, a meaningful representation of its application model must be defined and instantiated on the OPC UA server. This representation is realised in form of an OPC UA information model which follows a detailed mapping of information entities from the technology to be integrated (i.e., in this case the BACnet application model) to OPC UA nodes. A significant resemblance in BACnet and OPC UA facilitates this intention: both standards follow an object oriented approach. However, the modelling concepts in OPC UA are more advanced than in BACnet since, e.g., the latter does not support inheritance. Thus, defining a type hierarchy is not possible in BACnet. Another similarity exists in addressing the objects reflecting process data. In BACnet, objects have an `Object_Identifier`, properties have a `Property_Identifier`. In OPC UA, nodes are referenced by their `NodeId`. A mapping of these two addressing schemes is presented in Section 3.4.2. Furthermore, the concepts of services used to access data are similar in both standards. Access services to read and write exist in both worlds. Alarm and event handling are also defined which allow, for example, the monitoring of process variables and triggering of an event or an alarm if a change of values happens or a threshold is exceeded.

Due to the flexible modelling capabilities of OPC UA, the BACnet application model can mapped to the OPC UA object model in a relatively straight-forward way. The chosen approach is to transform BACnet objects to OPC UA complex objects. BACnet properties as members of BACnet objects are in turn mapped to OPC UA variables referenced by the corresponding OPC UA objects. In order to instantiate an entity in OPC UA, a type describing its structure has to be defined before. This needs to be done for the objects, variables, and references.

Since the value attribute of a variable is of a particular data type, the first step is to define a data type hierarchy that represents the available BACnet data types. Some of these BACnet data types can directly be mapped to the built-in OPC UA data types. For instance, the BACnet property type `REAL` (e.g., used by the property `Present_Value` of a BACnet `Lighting Output Object` type) can be modelled as the OPC UA `Float` data type. However, there are also complex BACnet property types that cannot be represented by built-in OPC UA primitive data types. Two examples thereof are the `BACnetObjectIdentifier` and the `BACnetObjectType`. However, the OPC UA base model defines the data type `Structure` intended to represent complex data types. These BACnet data types are modelled as subtypes of the OPC UA `Structure` data type. An exemplary part of this definition is shown in Figure 3.24.

All user-defined BACnet data types are subtypes of the user-defined abstract data type `BACnetPropertyDatatype` which is inherited from the OPC UA built-in data type `Structure`. For each user-defined structured data type, at least one encoding has to be defined that is used by clients to correctly interpret the user-defined data structure. In the proposed model, `DefaultBinary` encoding is chosen for all user-defined data types. For

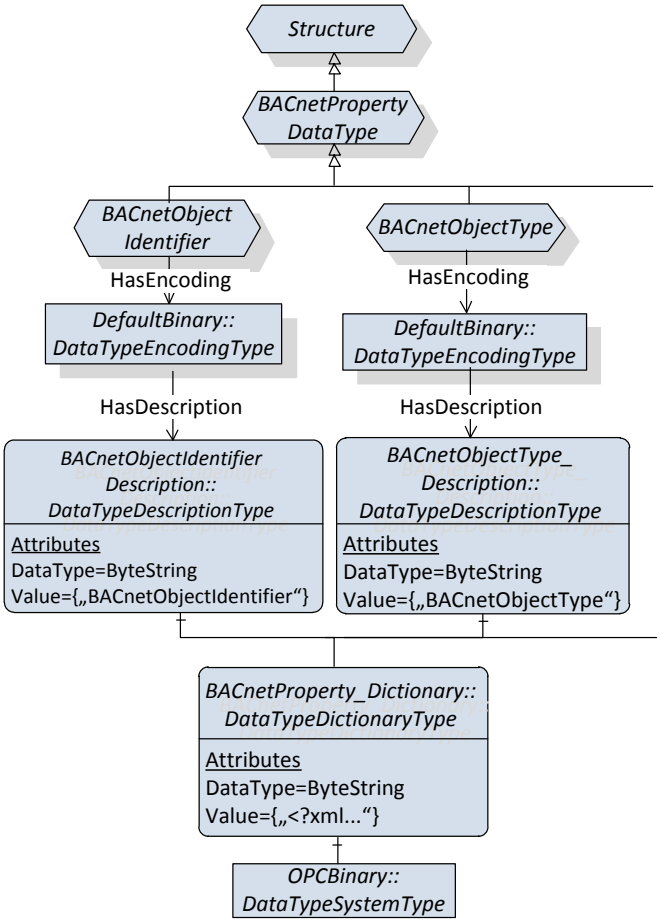

Figure 3.24: BACnet datatype definition

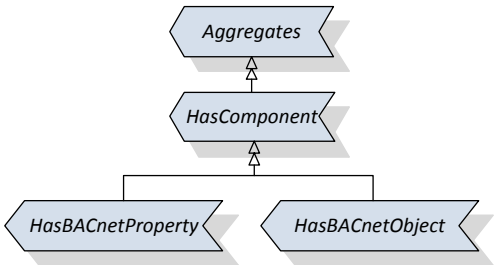

Figure 3.25: BACnet reference type definition

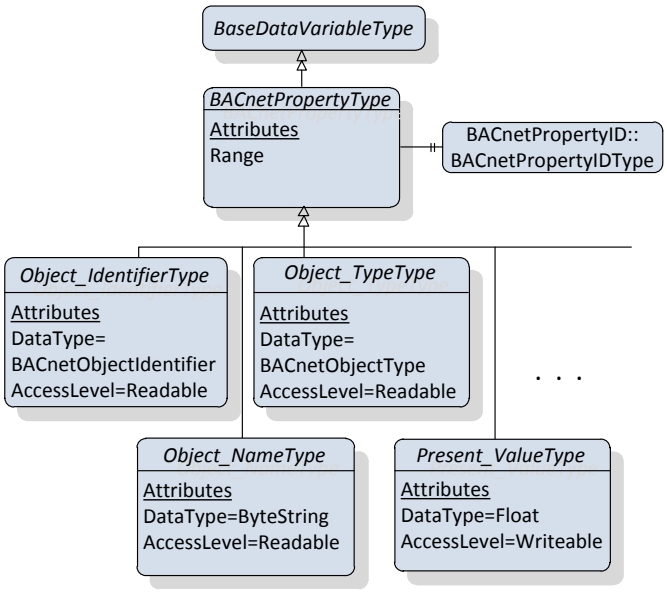

Figure 3.26: BACnet variable type definition

every encoding, a description of the type (represented by a `DataTypeDescriptionType` node) exists which in turn is a component of the `BACnetPropertyDictionary`. Within this user-defined dictionary, the entire encoding is described in XML format. For the BACnet property type `BACnetObjectIdentifier`, according to Part 3 of [32] the XML representation looks as follows:

```
<StructuredType Name="BACnetObjectIdentifier">
 <Field Name="ObjectType" TypeName="Bit" Length="10">
 </Field>
 <Field Name="InstanceNumber" TypeName="Bit" Length="22">
 </Field>
</StructuredType>
```

After having defined the BACnet data types, the BACnet properties have to be represented in OPC UA. To achieve this, user-defined OPC UA variable types are defined that are used for the instance declarations of the BACnet properties. Each of these BACnet specific user-defined variable types is a subtype of the abstract user-defined `BACnetPropertyType` variable type. This abstract variable type contains the user-defined OPC UA property `BACnetPropertyID` which represents the BACnet `Property_Identifier`. This attribute is unique for each BACnet property. A selection of these variable type definitions is shown in Figure 3.26. To create individual OPC UA variable types, the corresponding attributes of the new variable type have to be set. The `DataType` attribute is set to the corresponding user-defined OPC UA data type as described before. The `AccessLevel` informs the OPC UA client about access permissions to the particular variable. In this information model, the access permission facet of the conformance code in BACnet properties is mapped to the `AccessLevel` attribute each OPC UA variable type

has. Possible values are Readable and Writeable. The variable type definitions in Figure 3.26 reflect this circumstance. Examples for further attributes to be set are the BrowseName and the DisplayName which are both set to the human-readable name of the BACnet property defined in the standard. To assign the variables representing BACnet properties to OPC UA objects, references are used. To express the special semantics of these references, the new reference type HasBACnetProperty is defined. This reference type is inherited from the hierarchical type HasComponent as illustrated in Figure 3.25.

Now having all the necessary components available, the BACnet object types can be modelled in OPC UA. All BACnet object types are represented by user-defined OPC UA complex object types that are all subtypes of the abstract user-defined BACnetObjectType. This object type contains the BACnet properties Object_Identifier, Object_Name, and Object_Type which are common to all BACnet objects. The assignment of the variables that represent the BACnet properties to the corresponding object type is done by using the HasBACnetProperty reference mentioned before. To model the part of the conformance code of BACnet properties that specifies whether a property must be present or not, an OPC UA ModellingRule is defined for each variable. In this information model, only the Mandatory and Optional ModellingRules are used. The former forces the particular variable to be instantiated, the latter leaves only an option for that.

Inherited from the abstract BACnetObjectType, all BACnet object types that are specified in the standard can be represented in OPC UA. Figure 3.27 shows an example how the BACnet Device Object type and the BACnet Lighting Output Object type are defined using this concept. As shown in this figure, only the object specific variables are defined – the common ones are inherited from the supertype. As it is common in OPC UA, the HasSubtype reference is used to model the relation between sub- and supertype. An example of how meta information can be modelled is also shown in Figure 3.27 in form of the EngineeringUnit node referenced from the Power variable. To model the assignment of a unit to the value of a variable, the UPC UA built-in reference HasProperty is taken.

3.4.2 Instantiating and addressing of BACnet objects in OPC UA

After having presented how BACnet object and property types are modelled in OPC UA, it must be specified how instances of these types are represented by the OPC UA server and how the BACnet address information is mapped to OPC UA. In BACnet, each BACnet object is dedicated to exactly one BACnet device – BACnet objects are therefore never distributed across more than one BACnet device. Therefore, it is reasonable to use a device-centric view. Each BACnet device is represented as an OPC UA object instance of the user-defined object type BACnetDeviceType which in turn is a subtype of the standard OPC UA BaseObjectType (cf. Figure 3.27). The corresponding BACnet objects are assigned to the OPC UA object by using the user-defined HasBACnetObject reference which is a subtype of the standard OPC UA HasComponent reference type

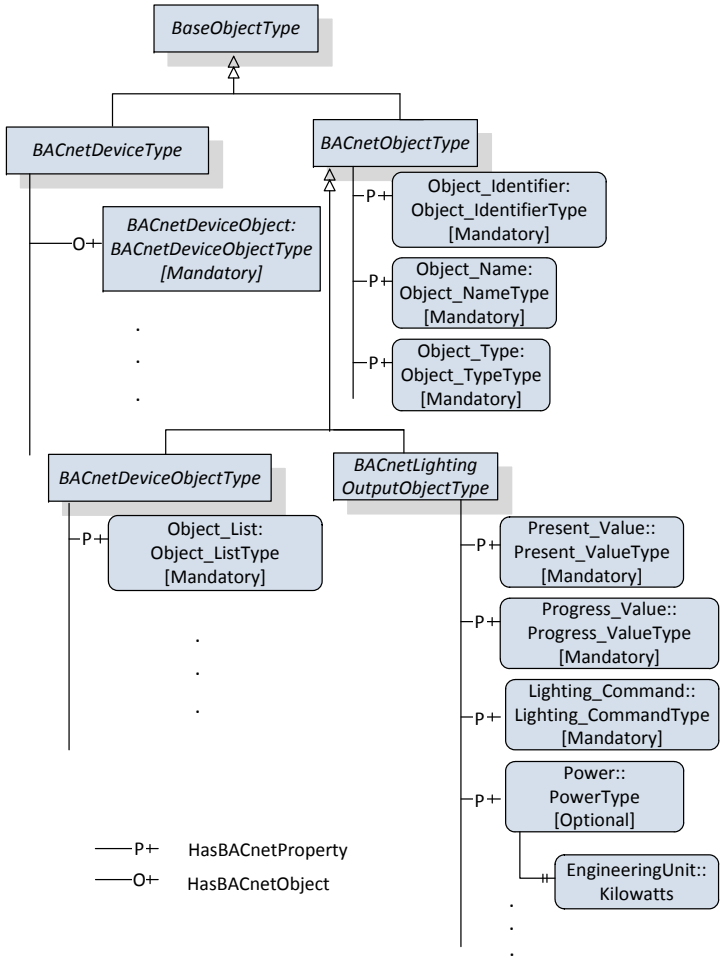

Figure 3.27: BACnet object type definition

(cf. Figure 3.25). Figure 3.28 shows an example how a BACnet device that contains a BACnet `Device Object` and a BACnet `Lighting Output Object` is modelled.

What is still remaining is how the BACnet properties can be addressed. BACnet properties are addressed by the `Property_Identifier` which is unique within the object. In the proposed information model, this can be done by reading the `BACnetPropertyID` property that is dedicated to each BACnet variable definition. To address the BACnet object, the `Object_Identifier` which is unique within the device is used. The `Object_Identifier` can be determined by the reading the value of the `Object_Identifier` variable that is mandatory for each BACnet object. Finally, to address the device itself, the BACnet `Device_Id` or the `Device_Name` which are both unique within the whole BACnet network can be used. To determine the BACnet `Device_Id` within the OPC UA model, the value of the `Object_Identifier` variable of the `Device Object` has to be read – to determine the `Device_Name`, the value of the

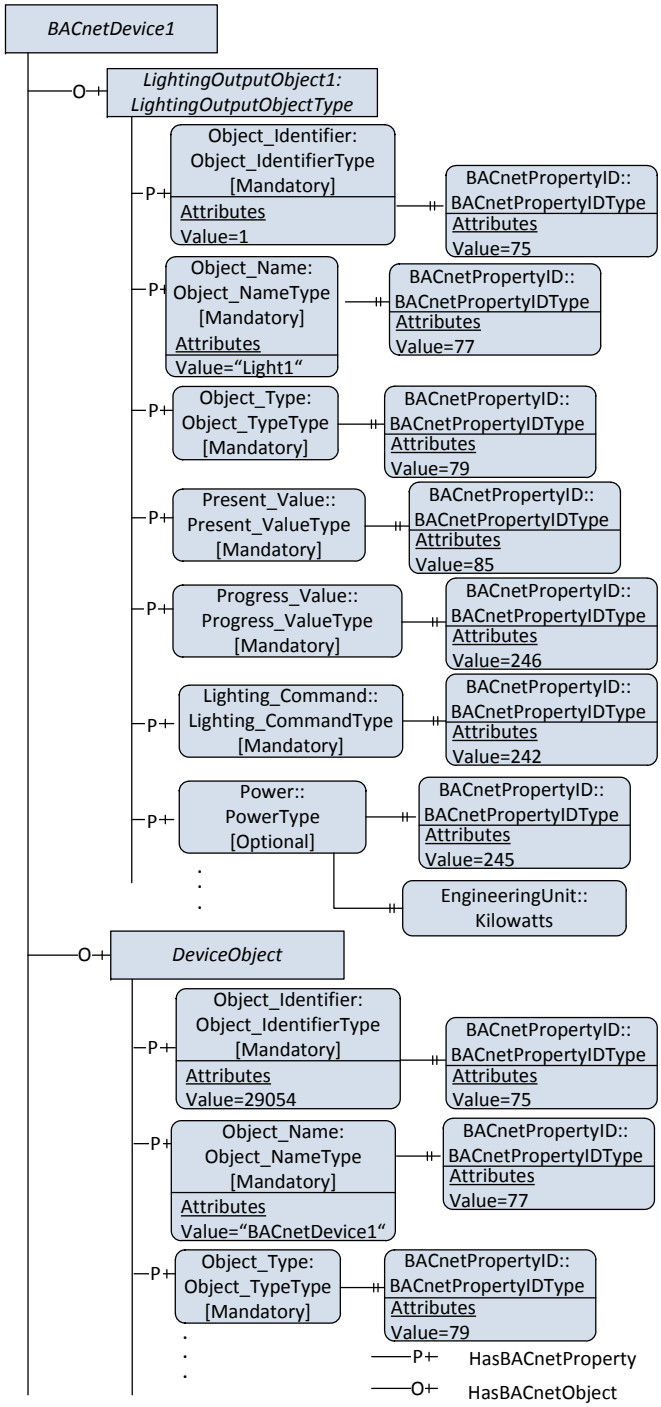

Figure 3.28: Instantiation of a BACnet device

Object_Name variable of the Device Object has to be retrieved. As a result, the combination of the value of the BACnetPropertyID property, the value of the OPC UA Object_Identifier variable, and the value of the Object_Identifier variable of the Device Object (or the value of the Object_Name variable of the Device Object) is used to address a BACnet property in the presented OPC UA model.

Figure 3.28 illustrates an instantiation of a BACnet Lighting Output Object. Consider, for example, an OPC UA client browses to the Present_Value variable of the BACnet Lighting Output Object and intends to read the value of it. To read its current value, the OPC UA server needs to invoke the BACnet ReadProperty service. To send this request, the BACnet address information has to be determined. First, the Property_Identifier is retrieved by reading the BACnetPropertyID property of the Present_Value variable (in the proposed example 85). Afterwards, the value of the Object_Identifier variable is read (in the given example 1). Then, the Device_Id is determined by reading the Object_Identifier variable of the Device Object (in the proposed example 29054). Using the combination of these values, the OPC UA server is able to send the ReadProperty request to the BACnet device. After having received the response, the OPC UA server is able to forward the present value to the OPC UA client.

3.4.3 Proof of concept

In the context of the research project "Web-based Communication in Automation (WebCom)"[6] an OPC UA framework called *Comet* has been developed by the project partner HB-Softsolution[7]. Among other software modules, it contains a Software Development Kit (SDK) for implementing Java based OPC UA servers. This server SDK is functionally separated into two parts: one is the core OPC UA server which is based on the OPC UA Java stack released by the OPC foundation. This core server loads the standard OPC UA information model plus any user-defined information model out of one or more XML files. This way the configuration part is completely isolated from the implementation's source code. As a result, the server's information model can be changed and extended even during runtime. The second part of the server module consists of a driver framework which allows to implement interfaces to particular network technologies that can be loaded into the core server. These drivers are responsible for interacting on protocol level with the targeted networking technologies. Depending on these technologies, the stack implementations can freely be chosen. The driver framework only provides an API for read and write methods which have to be implemented individually.

To evaluate the developed information model for BACnet, a proof of concept implementation was created. The implementation uses the Comet framework to implement an OPC UA for BACnet/IP networks. The driver implementation for the required interface to the BACnet/IP network is based on the open source *BACnet4J* stack[8]. It is

[6] http://www.webcom-eu.org/
[7] http://www.hb-softsolution.com/
[8] https://github.com/infiniteautomation/BACnet4J

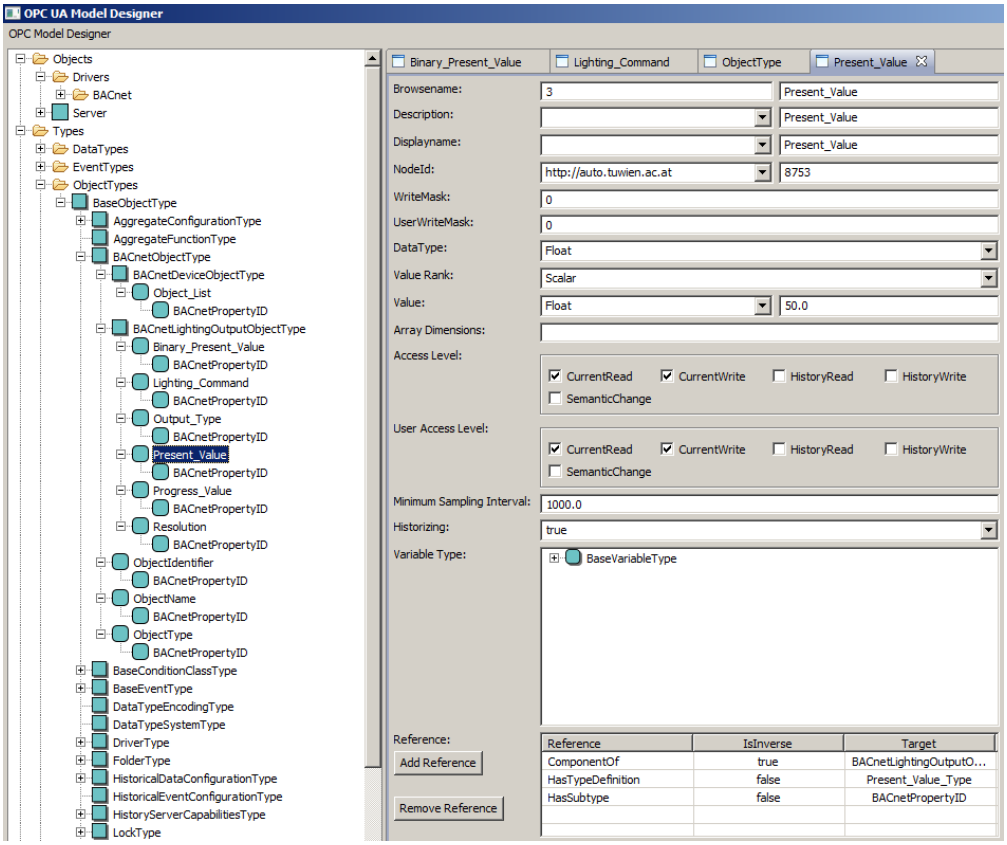

Figure 3.29: OPC UA Model Designer

a performant implementation of the BACnet/IP protocol supporting all BACnet services defined in the standard and the majority of objects. Emulating a BACnet device (i.e., creating a BACnet server) by instantiating local BACnet objects is also suitable for this implementation.

Another important software module of the Comet framework is the *OPC UA Model Designer* which is used to implement the developed BACnet information model. As an editing tool, it can be used to generate OPC UA information models and extend existing ones. It provides a Graphical User Interface (GUI) that supports the user in applying definitions of data types, variable types, reference types, and object types. Furthermore, instances can be derived from these type definitions in a very comfortable way. The hierarchical structure of the resulting information model is expressed by a tree view. A screenshot in Figure 3.29 shows the definition of the BACnet `Lighting Output Object` (without completeness of properties) embedded in its type hierarchy.

The information model created by the Comet Model Designer is finally exported in XML format which conforms to the XML Schema Definition (XSD) issued by the OPC Foundation [148]. This XML file can be opened again by the model designer for further

editing or it can be used as input for the Comet OPC UA Server.

To show the feasibility of the developed information model, an instance of a real-world BACnet controller was modelled within the Comet Model Designer. This BACnet controller is used to control a (HVAC) test installation within a laboratory setup. The resulting OPC UA model of this BACnet controller is loaded into the Comet OPC UA server. Tests have proven that it is possible to control the HVAC test installation using a standard OPC UA client (UaExpert from Unified Automation[9]) connected to the BACnet/OPC UA server.

3.5 Integration of building and industrial automation domains

Bridging the two domains of building and industrial automation systems at their management levels is a topic hardly regarded so far. Nevertheless, cross-domain integration promises significant benefits in terms of resource consumption and costs optimisation where systems of both worlds encounter each other. To this aim, this section presents an OPC UA information modelling framework providing a holistic information base for both domains. This facilitates the implementation of cross-domain management applications also in the context of smart grids with the potential goals to coordinate the reuse of waste energy, to avoid peak loads and to predict energy demand.

In the past, OPC UA models have been defined to integrate automation system within a distinct domain, whether building or industrial automation. Going one step further, the same can be done when creating a common view on these two different domains. A scenario where this is applicable appears when considering a factory plant with an associated office building nearby. Normally, when a distinct size is exceeded, functional buildings are highly automated. In the industry, automation systems are also very common nowadays. By integrating the automation technologies of both domains into one unified management system, several benefits arise. Based on a condensed information base, cross-domain optimisation tasks become possible regarding the overall resource consumption, for example. In times where an increasing awareness on resource consumption and sustainability is established, this is an important goal to be met. Beyond that, peak load avoidance and resource demand prediction are applications where a cross-domain integration model can act as a basis, too. Both are requirements for a future smart grid roll-out. These goals can be achieved when a comprehensive management application (like a rule-based system, which is out-of-focus here) can be put on top of this cross-domain integration approach. It hereby gains unified access to the underlying automation networks. The integration layer acts as a well-structured source for process data and as an enabler to realise the mentioned use case. In the management application, predefined rules can be processed by an inference engine to derive optimised control decisions as well as demand predictions based on the information provided by this model.

[9] https://www.unified-automation.com/products/development-tools/uaexpert.html

In a preliminary publication where the author contributed[149], a comparison of different methods for integration has been carried out. This resulted in an information modelling approach based on OPC UA. First steps towards a modelling framework with the goal to address cross-domain management application has finally been introduced. These concepts are refined here where the focus lies on the description of the two-tier modelling approach. This included information models dedicated to distinct automation technologies on the one hand and on the other hand a cross-domain data aggregation model. The results of an empirical evaluation by means of a prototype implementation are shown in the last section of this chapter.

3.5.1 Cross-domain scenarios

The methodology applied in this work starts with an exploration of representative use cases with a resource-saving background using information from both industrial and BASs.

Energy efficiency in production plants is a topic getting little attention, yet. In many cases, e.g., in heavy industry plants, cooling water is disposed without considering its residual thermal energy. The fictive "Company XY" which will be taken as an example here processes steel in rolling mills. A huge amount of heat needs to be purged from the rollers to prevent them from overheating. Nearby the factory building, there is an office building housing the company's administration. This building needs to be heated in winter and air-conditioned during summer time. It is obvious that waste energy from the rolling mill, if available, can potentially be used for supplying the heating of the administration building and also the air conditioning system by using an absorption chiller. Besides the physical infrastructure necessary to transport hot water from the plant to the office building and distributing it properly to the consumers, HVAC automation networks and the control systems in the factory plant must negotiate the waste energy exchange in some way.

Not only when it comes to smart grid integration [150], [151] peak-load avoidance is a remarkable topic. It is also an effective way to save energy costs [152]. When considering building automation and the factory automation systems in a holistic way, peak loads in the factory plant can be compensated by reducing power of the HVAC system. The electricity intake of an air conditioning system for a medium-sized building can be remarkable. The huge thermal capacity of such a building will almost completely absorb a short-term break in cooling supply. Like in the previous scenario, information exchange must take place between the two domains - factory and building automation.

Inferences about the current operating condition of equipment can be drawn by monitoring their power consumption. This method, which is called condition monitoring, allows, for example, the in-time detection of bearing wear-outs or failed light bulbs. Ideally, failures are indicated to the facility manager or the plant operator which are then enabled

to trigger countermeasures like issuing maintenance personnel.

From these use case scenarios, requirements with respect to information necessary to enable the realisation of these use cases are derived. This is done by evaluating the following two questions:

- Which information might be exchanged between building and industrial automation systems?
- Which information is common to both domains?

As a result of this process, the following pieces of information originating in both domains are necessary to achieve the proposed functionality:

- Measurement values of electricity consumption, heat consumption or dissipation
- Pre-processed status information about the operating conditions of equipment
- Setpoints to feed control decisions back to the automation networks
- Topological information of buildings and plants to uniquely locate equipment

3.5.2 Two-tiered OPC UA information model

The modelling approach shown here shall provide a holistic view on cross-domain automation systems. As a result of this use case analysis, not only functional information needs to be represented, but also a topological description of the physical properties and the equipment deployed is necessary. The resulting topology model scales from a fine-grained device oriented view to an abstract exposition of aggregated information dedicated to distinct use cases. This reflects the intended use of this information model in a two-tier OPC UA server hierarchy. The lower level OPC UA servers integrate distinct automation technologies from a specific domain, whereas the upper level OPC UA aggregation server integrates the information from the underlying ones and provides the desired abstract, domain-independent view.

Topological model

This part of the information model has already been introduced in [149]. Figure 3.30 shows the definition. It basically follows the ISO 16484-3 [153]. Therefore, and also because it is quite self-explaining we will not go into it very deeply. However, important characteristics of it are the connection points for the device oriented technology models based on the DI specification [145] on the one hand and for the aggregation model on the other hand. So to speak, the topological model is the central element, the common denominator of the two tiers of the model. It contains `TopologyElementType` InstanceDeclarations referenced by each topological object. The `TopologyElementType` ObjectType is defined in the DI specification. It represents the “basic information components for all configurable elements in a device topology”. This includes, most important, a

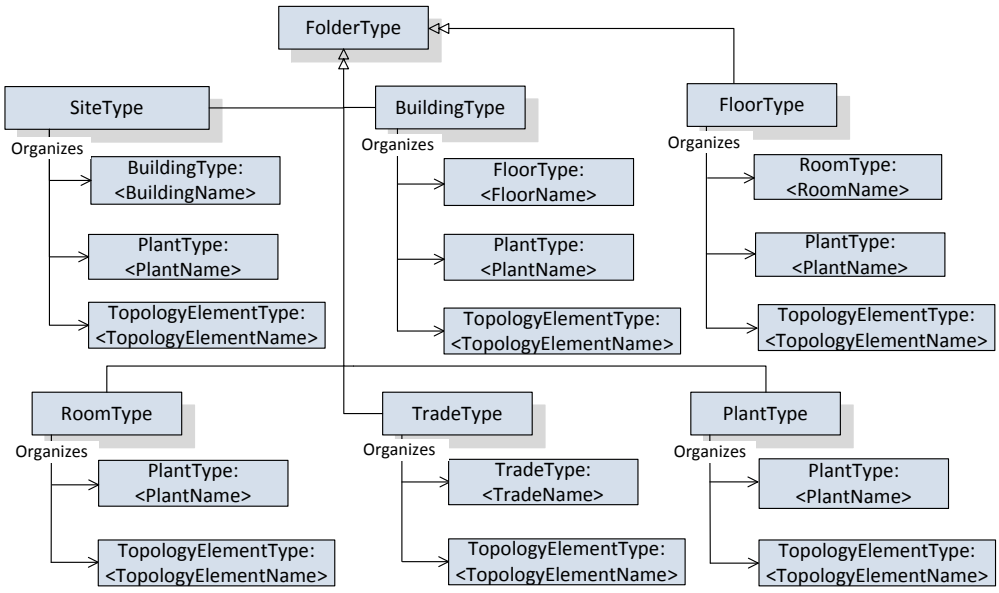

Figure 3.30: Topological model

`ParameterSet` and a `MethodSet` object, an object to group parameters and methods by their semantic and a locking mechanism for synchronisation purposes. Moreover, the `TopologyElementType` is the super type of `DeviceType` and the `BlockType` ObjectTypes also defined in the DI specification. This means for the proposed topology information model that the `TopologyElementType` InstanceDeclaration acts as a placeholder for `DeviceType` and `BlockType` InstanceDeclarations. This circumstance is later used when integrating automation technology models (based on the `DeviceType`) and the aggregation model (based on the `BlockType`) into the topological model.

Technology models

In the meanwhile, a number of so-called Companion Standards for OPC UA exist, e.g., the already mentioned DI, the OPC UA Information Model for IEC 61131-3 [154] (OPC UA Information Model for IEC 61131-3 (PLCopen)) or the OPC UA Information Model for BACnet [56] which has currently the status of a release candidate. These additional specifications built upon the OPC UA base information model aim at vertical integration of automation technologies as an approach to overcome the well-known interoperability issue.

In order to show the integration of BASs to the proposed cross-domain approach, again KNX is taken as a prominent technology representative. Since this cross-domain modelling framework based on the DI standard [145], this is also a prerequisite for the technology information models of the two domains to be integrated. To be precise, this restriction is a result of the use of the `TopologyElementType` as a linking element for technology-specific models in the topological model described in the previous section. Therefore, the

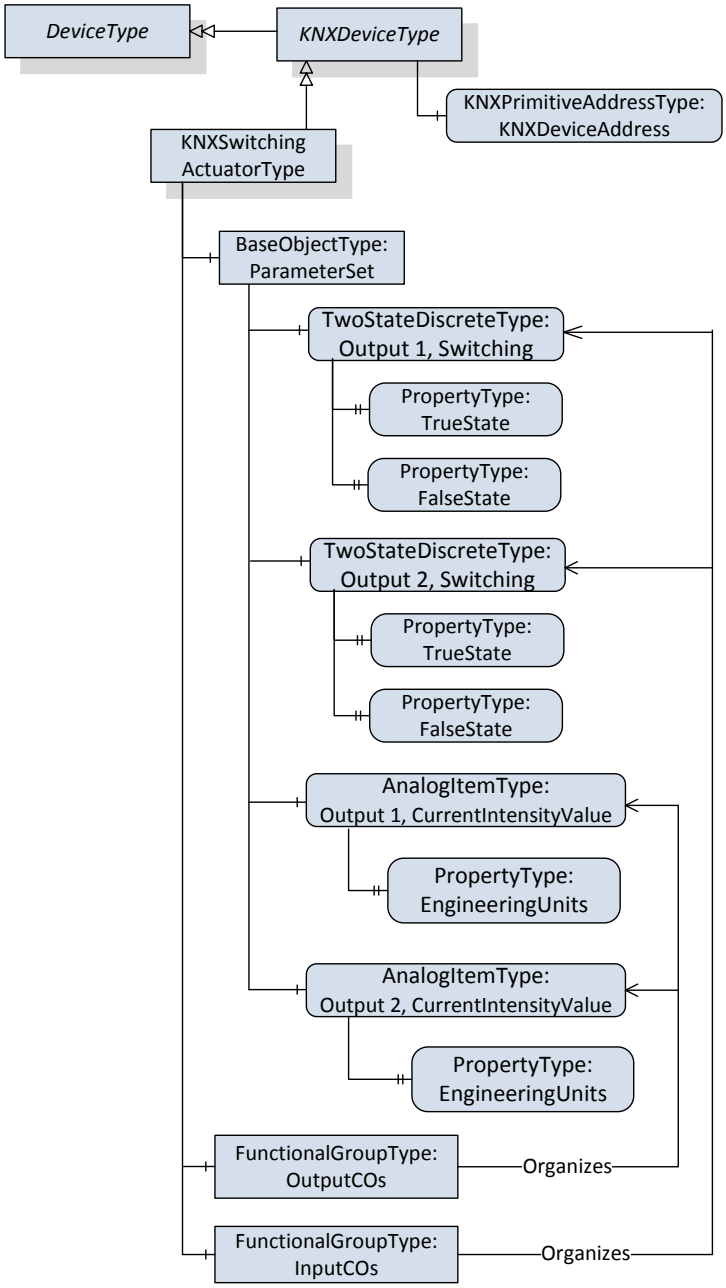

Figure 3.31: OPC UA information model for a KNX switching actuator (refined)

device view of OPC UA information model for KNX presented in Section 3.3, is redefined in a way such that it builds upon the device model of the DI specification. The resulting model of an OPC UA ObjectType definition for a KNX switching actuator is shown in Figure 3.31. The introduced type hierarchy starts with an abstract `KNXDeviceType`, which is a subtype of the DI `DeviceType`. The Property KNXDeviceAddress models the physical address of the KNX device. As a child of the `KNXDeviceType`, a concrete `KNXSwitchingActuatorType` is defined. As required by the DI model, it references a `ParameterSet` InstanceDeclaration to group the devices' COs. `TwoStateDiscrete-Type` and `AnalogItemType` VariableTypes as defined in the OPC UA Specification Part 8: Data Access [146] are taken to model the actual COs. In order to express a categorisation in `InputCOs` and `OutputCOs`, the concept of `FunctionalGroups`, which is also taken from the DI specification is used.

As a representative of an information model for an industrial automation standard, the PLCopen information model, also based on DI and therefore compatible with the topological model, shall be mentioned in this context. It exposes configuration data as well as runtime data (variables of an IEC 61131-3 PLC program) via OPC UA. It therefore allows to make relevant information of an IEC 61131-3 compliant PLC accessible via OPC UA. This will be shown later in one of the cross-domain model instances and also used in the course of the empirical evaluation.

Aggregation model

The information aggregation part of this cross-domain model is designed to specifically expose datapoints necessary to realise the use cases previously stated. This is in contrast to the designation of the similarly named OPC UA Specification Part 13: Aggregates [155] which defines an information model for exposing pre-processed historical data. Interpolated and averaged data, extreme values and results of statistical calculations can be integrated using the meta model of this part of the UA specification.

For the aggregation model defined in this work however, types from the DI specification are used as a foundation for the new types introduced here. The definition is illustrated in Figure 3.32. Inherited from the `BlockType`, which is a subtype of the already familiar `TopologyElementType`, the `ConsumptionMonitoringBlockType`, the `WasteEnergyMonitoringBlockType`, the `RunControlBlockType`, and the `ConditionMonitoringBlockType` are defined. In order to improve structuring, the `AggregationBlockType` is introduced. Instances of this type are intended to be finally integrated into the topology model. The `AggregationBlockType` further references use case specific BlockTypes as its Components whereas the `OptionalPlaceholder` ModellingRule expresses a cardinality of zero to arbitrary many possible instances of each BlockType. The `RevisionCounter` Property each of the five ObjectTypes introduced here have in common is inherited from the definition of the `BlockType`.

The `ConsumptionMonitoringBlockType` and the `WasteEnergyMonitoring-`

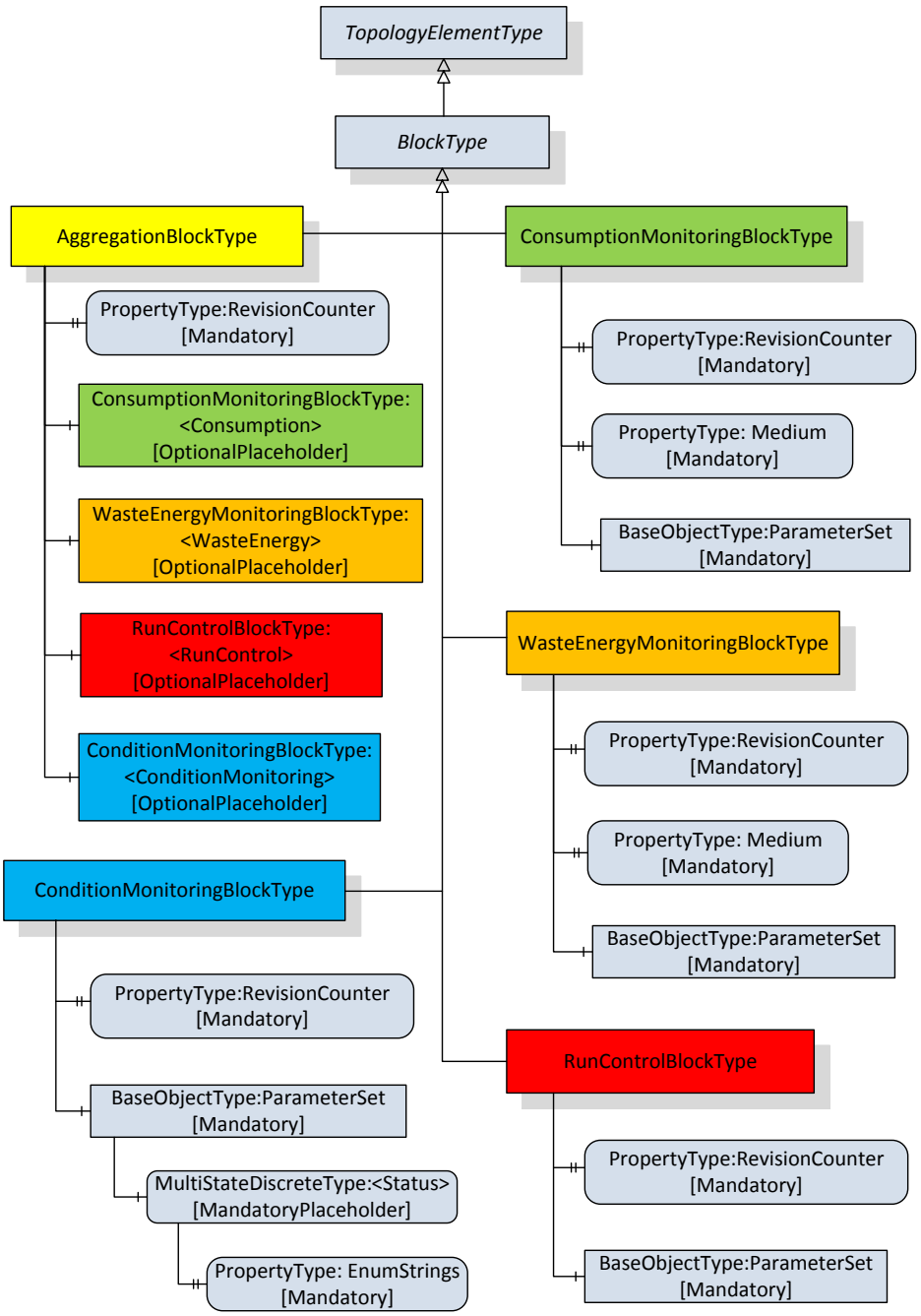

Figure 3.32: Aggregation model

BlockType indeed have the same properties but clearly differ in their semantics. The former BlockType describes the consumption of energy of a distinct carrier indicated by the Medium Property, whereas the latter does the same for emitted energy. The ParameterSets will be filled with actual datapoints when the BlockTypes are instantiated.

The ConditionMonitoringBlockType differs in one aspect from the other BlockTypes. In the ParameterSet InstanceDeclaration of this BlockType an InstanceDeclaration of a Status variable is already present. This MultiStateDiscreteType InstanceDeclaration, defined in OPC UA Part 8: Data Access [146] uses a MandatoryPlaceholder ModellingRule. This means that the ParameterSet of an instance of this BlockType must have at least one Status variable. The MultiStateDiscreteType requires an EnumStrings Property defining the different values a variable of this type may have. Examples for possible values are Normal, Critical, and Failed. It is task of the aggregation server to classify the analog value under observation (like the current consumption of a lighting) by comparison to predefined thresholds.

In order to allow a superior layer (e.g., a Knowledge-Based System (KBS)) to feed back the results of its control decisions, setpoints from underlying automation systems shall be exposed by the RunControlBlockType in this information model. Its ParameterSet object shall be taken to reference analogue setpoints for temperature values, e.g., or binary and multistate values for representing run levels of machines.

The information model in Figure 3.33 exemplarily illustrates the instantiation of elements from the aggregation model as well as from the topological model. Therefore, the term *topo-aggregation model* is introduced. The topological elements describe the structure of a company site, CompanyXY, including an office and a factory building. The office building has several floors, where two of them are shown in this figure. The second floor includes a conference room where a KNX lighting application is deployed (not visible in the aggregation model). The companies factory building houses a rolling mill consisting of a number of rollers (one of them is shown in this figure). Several AggregationBlockType instances are referenced by elements of the topological model. For clearance, the actual datapoints exposed by the aggregation blocks are not shown in this model. The mapping of datapoints from technology models to the topo-aggregation model is illustrated in the following section.

The OfficeBuldingMonitoring aggregation block provides sub-blocks which aggregate metering data for heat and electricity consumption of the office building as well as setpoints for the HVAC system. For a more fine-grained monitoring of the energy consumption of a distinct floor, aggregation blocks can be applied for a specific floor or even a room, like shown for the Floor2Monitoring and the ConferenceRoomLightMonitoring blocks. Metering data at building and floor levels can be gained, for example, from smart meters. The ConferenceRoomLightMonitoring block provides besides information about the electricity consumption also condition monitoring information gained from measurement values of a KNX light switching actuator. At the production plant branch of the topological model hierarchy, FactoryBuildingMonitoring and Roller1Monitoring aggregation blocks are instantiated. The former fulfills the same

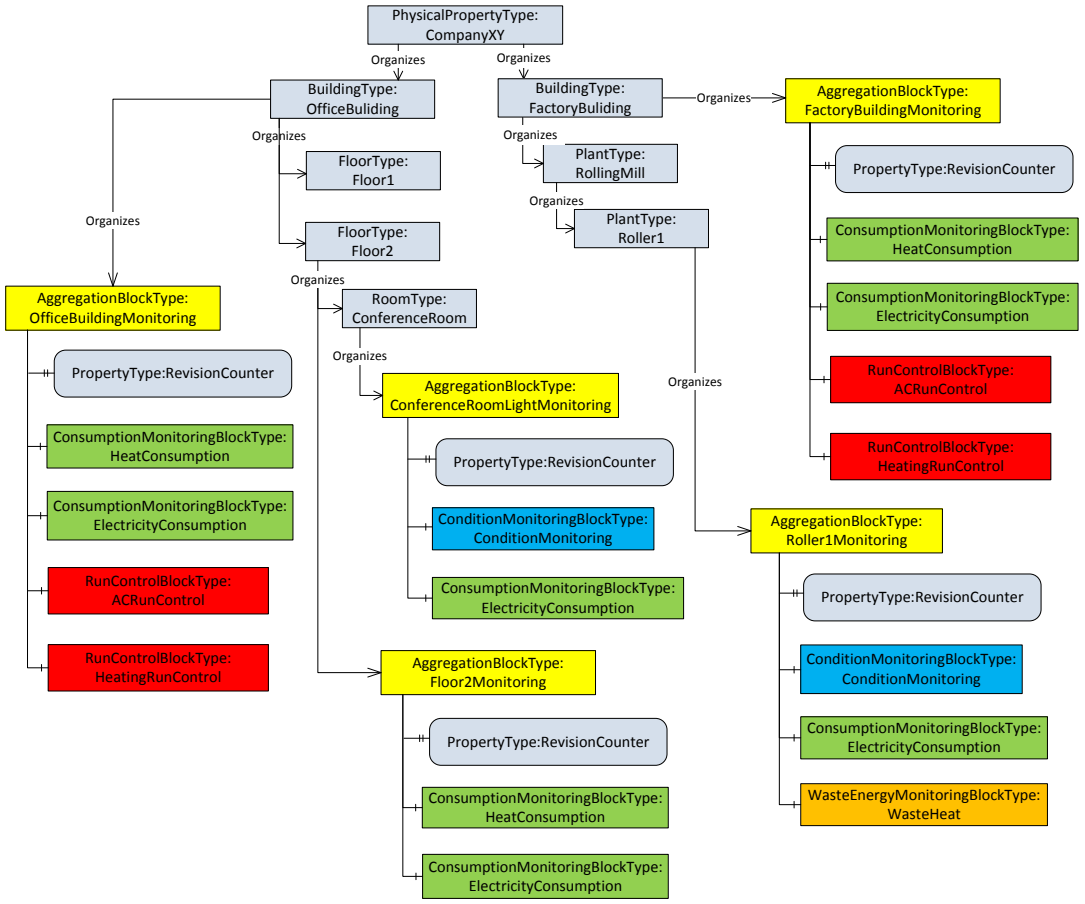

Figure 3.33: Topo-aggregation model instance

purpose as the `OfficeBuldingMonitoring`. The `Roller1Monitoring` block not only provides condition monitoring and electricity consumption information about the electrical drives of the roller but also informs about the amount of waste heat available originating in its cooling system.

Model transformation

As previously described, the modelling approach presented in this cross-domain integration approach bases on a two-layered abstraction. The upper layer (the topo-aggregation model) has already been discussed in the previous section. The lower layer is constituted of technology specific OPC UA information model instances on the one hand, for example the one for KNX and IEC 61131-3, and on the other hand it includes instances of the topological model. Therefore, this lower layer is called *topo-technological* model. The transition of datapoints from the topo-technological model to the topo-aggregation level

is defined by the following rules:

- *Rule 1*: The `ParameterSet` of instances of the `MeteringBlockType`, the `WasteEnergyBlockType` and the `RunControlBlockType` directly reference variables from technology models using a HasComponent reference
- *Rule 2*: The `ConditionMonitoringBlockType ParameterSet` has one or more status variables of the `MultistateDiscreteType`. No direct mapping of datapoints originating in a technology model takes place here. Instead, the values of these multistate variables are calculated by the aggregation server. Analog values exposed by technology models are classified by comparison against predefined thresholds. Possible results of these calculations can be, e.g., `Normal`, `Critical` and `Failed`.

The application of these rules is exemplarily shown in Figure 3.34 for a KNX BAS as well as in Figure 3.35 for an IEC 61131-3 model. Right-hand in Figure 3.34, a topo-technological model of the conference room in the fourth floor equipped with a KNX switching actuator is presented. The runtime datapoints (actually, the COs) of this device are exposed beyond the `ParameterSet` object. They include two binary setpoints, `Output 1, Switching` and `Output 2, Switching` of the `TwoStateDiscreteType`. Their `TrueState` and `FalseState` properties provide the meaning of the two binary states, namely in this case "On" and "Off". A `FunctionalGroup` object categorises the two setpoints as `InputCOs`. The `Output 1, CurrentIntensityValue` and the `Output 2, CurrentIntensityValue` act as `OutputCOs`. They provide feedback of each light circuits' electricity consumption. KNX devices with this feature already exist on the market[10]. `EngineeringUnit` properties inform about the unit of these physical quantities.

In the red-boxed part (the topo-aggregation model) of Figure 3.34, the `ConferenceRoomLightMonitoring` aggregation block is shown in detail. The `ParameterSet` of the `ConditionMonitoring` sub block exposes two multistate variables, `Circuit1Status` and `Circuit2Status`. According to *Rule 2*, the values of these variables are calculated by the aggregation server classifying the values of the `Output 1, CurrentIntensityValue` and the `Output 2, CurrentIntensityValue` analog items of the KNX switching actuator model. Concerning the `ElectricityConsumption` block type, information aggregation is performed by applying *Rule 1*. Its `ParameterSet` object is linked to the two current consumption variables of the KNX switching actuator model by `HasComponent` references and therefore provides direct access to these measurement data at aggregation level.

Analogously to the previous model transformation example, Figure 3.35 in its blue-boxed part shows the OPC UA information model of an IEC 61131-3 compliant PLC. In this example, `PLC1` is dedicated to control (parts of) `Roller1` of our `RollingMill` and is

[10] GIRA http://www4.gira.com

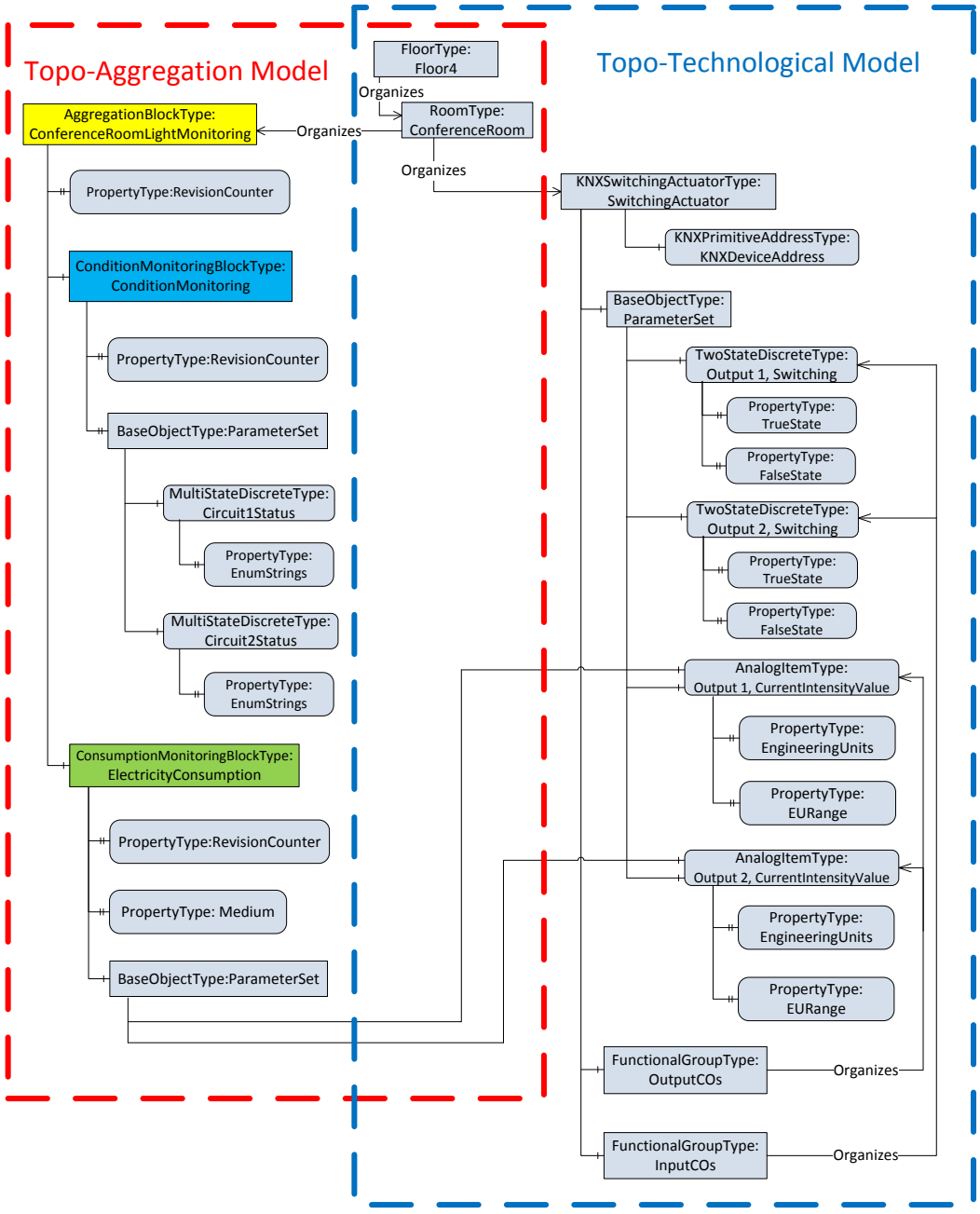

Figure 3.34: Datapoint transition in a building automation model

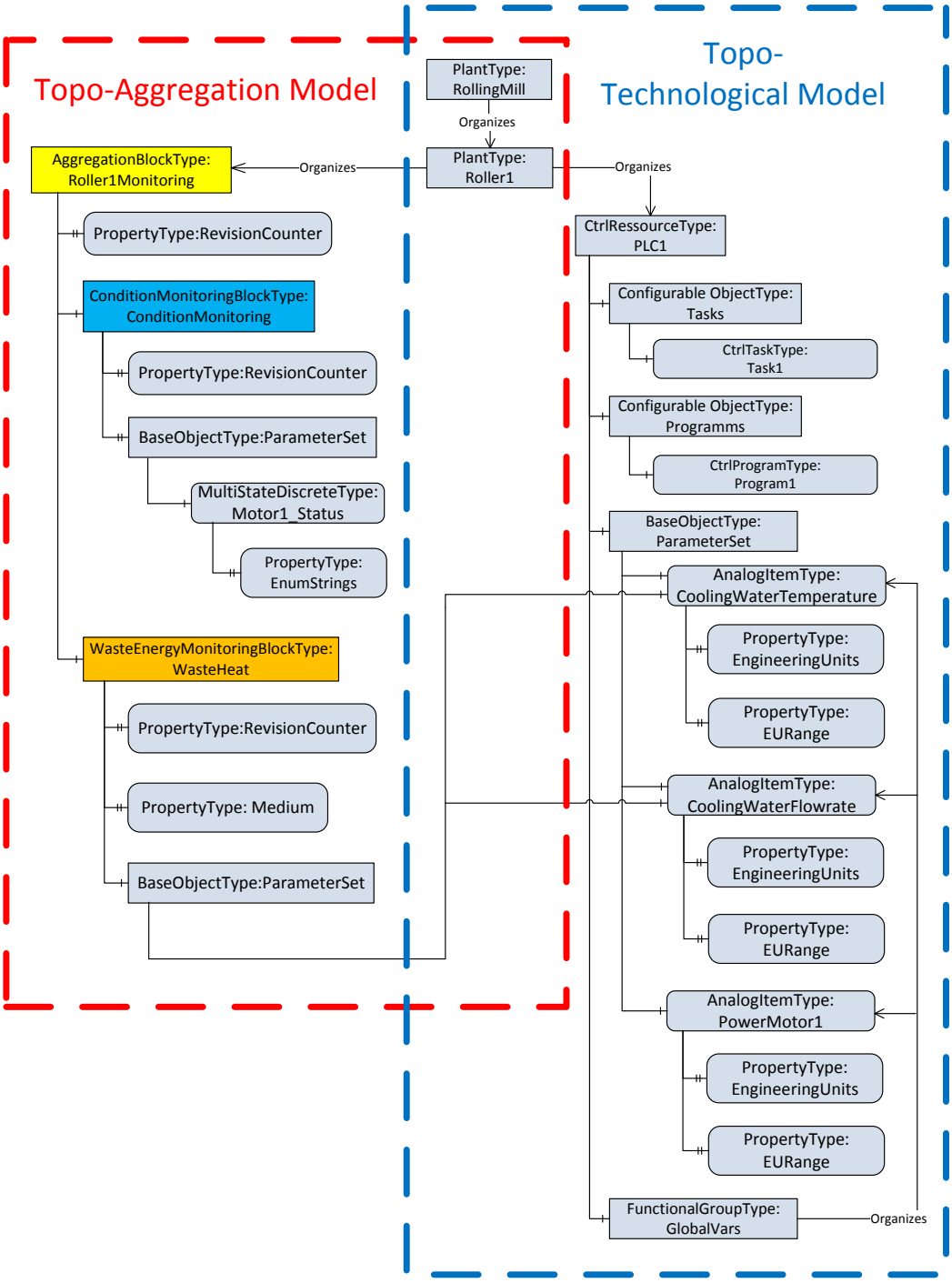

Figure 3.35: Datapoint transition in an industrial automation model

therefore referenced by the according topology items. Besides some static information, the OPC UA server running on the PLC provides access to a number of global variables of its IEC 61131-3 program (indicated by the `GlobalVars` functional group). In this case, the `CoolingWaterTemperature`, the `CoolingWaterFlowrate`, and the `PowerMotor1` are exposed. The topo-aggregation model in Figure 3.35 consists of an `Roller1Monitoring` aggregation block referenced by the `Roller1` object. The status information (`Motor1Status`) provided by the `ConditionMonitoring` block is calculated based on the value of `PowerMotor1` using *Rule 2*. Depending on deviation from a predefined normal value, failures in this drive and the attached mechanics can be detected. The `WasteHeat` block is intended to expose information about the amount of waste heat currently available by the roller's cooling water. Therefore, the `ParameterSet` of this block references the `CoolingWaterTemperature` and the `CoolingWaterFlowrate` variables of the PLC's information model. By this means, these physical quantities including engineering units are transformed to aggregation level. An OPC UA client operating at the top level of a cross-domain automation system can easily derive the amount of waste energy available and is enabled to base its control decisions on heat distribution upon this information.

3.5.3 Proof of concept

In this section, a prototypical, distributed implementation of the model approach described in the previous sections is presented. The two-tiered model hierarchy is also reflected by the OPC UA server hierarchy of the prototype. As presented in Figure 3.36, a server architecture divided into two layers is set up. The lower layer servers integrate the components of the underlying automation networks by instantiating the respective topo-technological models. The aggregation server on top concentrates relevant information from the sub servers and instantiates the topo-aggregation model.

For the factory automation domain, a Fischertechnik[11] tool machine acts as a target model located at TU Dresden. Its fieldbus devices are connected to an *isNet Lite* by *ifak system*[12]. The isNet Lite is a general-purpose Ethernet-gateway which is capable of integrating a number of fieldbus protocols. This is enabled by a modular design allowing to connect up to five different fieldbus modules to the isNet Lite. In this setup, a Profibus DP module (*isNet DP*) is used to integrate a number of Profibus devices. The integration platform runs an OPC UA server which is able to expose datapoints of the underlying field devices to superior applications. The OPC UA information model the isNet Lite implements is compliant to the OPC UA for IEC 61131-3 specification.

At TU Vienna, a process model emulating a central air conditioning system common in functional buildings is set up. It consists of a variety of field devices necessary to measure and influence the air flow, like fans, pumps and flaps as well as sensors for temperature and humidity positioned in the air flow. This process model also includes a

[11] www.fischertechnik.de
[12] www.ifak-system.com

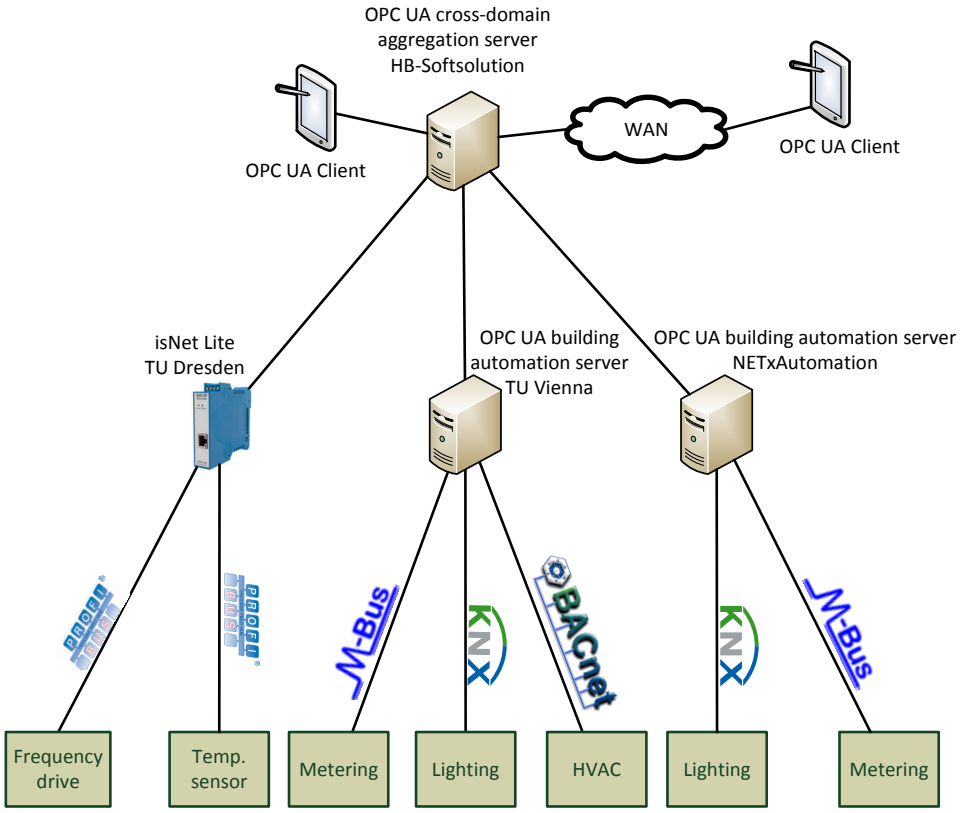

Figure 3.36: Two-tier server architecture

test chamber where the conditioned air is fed in. The sensors and actuators are connected to a Siemens *PXC100* BACnet controller. Via a dedicated I/O module, this controller is also able to integrate M-Bus metering devices by mapping the meter readings to the BACnet application layer, i.e., BACnet objects. In our setup, we have a heat meter whose cumulated energy datapoint is mapped to a BACnet Analog Input object. For lighting control, a simple KNX installation consisting of several switches, dimming sensors and actuators is deployed.

By means of an OPC UA server which is based on the Java server SDK offered by *HB-Softsolution*[13], these three technologies are accessible via OPC UA. The respective OPC UA technology models and driver interfaces for the server have already been introduced in the previous Sections 3.2, 3.4. One additional OPC UA server in this setup dedicated to the building automation domain has been implemented by NETxAutomation[14]. This server is currently able to integrate KNX, M-Bus and several other building automation technologies. For the implementation of the OPC UA aggregation server at the top level

[13] www.hb-softsolution
[14] www.netxautomation.com

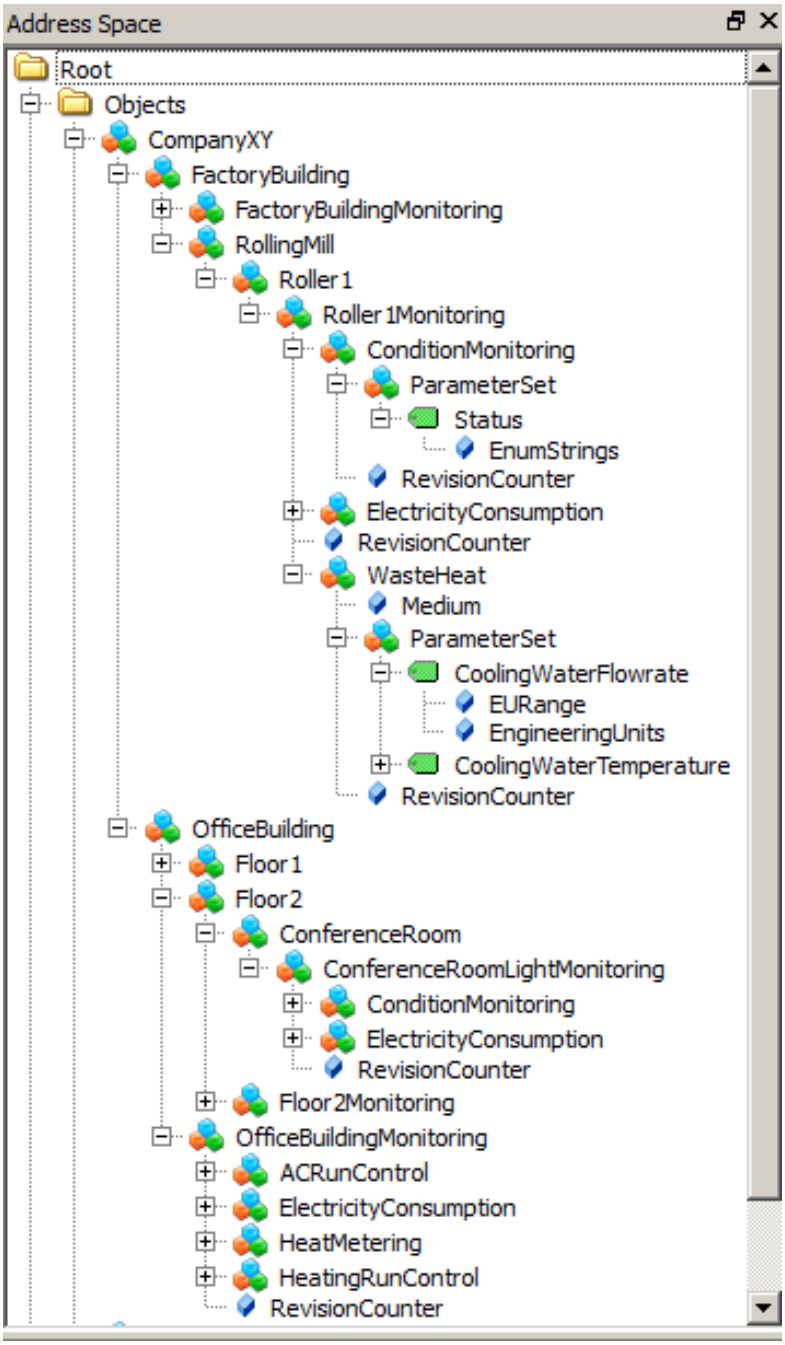

Figure 3.37: OPC UA client view

of this hierarchy, the Java Server SDK from HB-Softsolution was taken. In order to connect to the underlying OPC UA technology servers, the aggregation server needs to be equipped with a client interface. The address space of the server part holds instances of the introduced topo-aggregation model. It heavily depends on the actual use case and the current application which datapoints of the technology models are desired for aggregation and further, via which aggregation block they should be made accessible. Therefore, this needs to be decided individually in each case. In a first step, the required BlockType instances are loaded into the aggregation server address space. This is facilitated by using a GUI-based OPC UA modelling editor called *ModelDesigner* which is also provided by HB-Softsolution. It generates an XML output representing the information model which can be loaded by the aggregation server during runtime. In a second step, the model transformation introduced in Section 3.5.2 is put into practice. Instances of both technology specific OPC UA models and the aggregation model present in the address spaces of the technology servers and the aggregation server are logically linked. Like shown in the model transformation section, references fulfilling this purpose need to be instantiated. For this purpose, a graphical tool for configuring the aggregation server was developed. Via its GUI it allows the user to enter the NodeIds of the respective nodes of the aggregation and the target technology model. This results in an according reference across both respective servers to be created. However, not only direct references like necessary to apply transformation *Rule 1* (defined in Section 3.5.2) are possible. Also arithmetical and logical calculations as well as condition checks can be executed on the originating value. Hereby, the implementation of transformation *Rule 2* is supported by the aggregation server as well. Furthermore, even more complex operations can be carried out by loading external JavaScript files containing the desired instructions.

Figure 3.37 shows the address space of the aggregation server exposed to an OPC UA client. This screenshot was taken from *UAExpert*, a freeware client software of Unified Automation[15]. It can be seen that the topo-aggregation model instance proposed in Figure 3.33 was implemented for this prototype. Going into detail of the `Roller1` object, the `Roller1Monitoring` aggregation object groups a number of blocks. One of them, the `WasteHeat` block exposes two variables via its `ParameterSet`, the `CoolingWaterFlowrate` and the `CoolingWaterTemperature`. These two datapoints originate in the IEC 61131-3-compliant address space of the isNet Lite. This shows the application of model transformation *Rule 1*. *Rule 2* is applied concerning the `Status` variable of the `ConditionMonitoring` block. Its multi-state value is calculated by the aggregation server based on the power consumption value of the roller drive, which is made accessible via the `ElectricityConsumption` block.

3.6 Results

To the aim of first level integration, OPC UA information models have been presented for the prominent building automation technologies KNX, BACnet and also for M-Bus

[15] www.unified-automation

to cover the integration of smart meters. The KNX information model also includes meta information about the physical location of a KNX device, the associated trade as well as the network topology. In order to enable an OPC UA server to reflect the actual configuration of a live KNX installation, a method has been introduced to automatically import the ETS4 XML export data into an OPC UA server. To extend interoperability to systems from the domain of industrial automation, an OPC UA information model for bridging the fields of building and industrial automation systems has been defined. Starting from the previously defined automation technology mappings to OPC UA and a basic topological model describing buildings and plants, transformation rules towards this cross-domain aggregation model have been defined. Following the requirements derived from representative use cases, the application of these transformation rules is shown by means of integrating datapoints from OPC UA information models for KNX and IEC 61131-3 into the cross-domain model. In order to show principle practical operation, prototypical implementations were described for BACnet, KNX and cross-domain integration.

CHAPTER 4

Second level integration into knowledge-based systems

This chapter is based on the contributions of the author to the following publications:

A. Fernbach, W. Granzer, and W. Kastner,"Knowledge Based Building and Energy Management," *Tagungsband der e-nova 2015, Forschungs- und Studienzentrum Pinkafeld*, Nov. 2015.

A. Fernbach, I. Pelesic, and W. Kastner, "Linked Data for Building Management," *Proc. of the 42nd Annual Conference of the IEEE Industrial Electronics Society (IECON 2016)*, Oct. 2016.

A. Fernbach and W. Kastner, "Gebäudemanagement durch wissensbasierte Systeme," *Kommunikation und Bildverarbeitung in der Automation: Ausgewählte Beiträge der Jahreskolloquien KommA und BVAu 2016 Zum 10jährigen Jubiläum des InIT - Institut Für Industrielle Informationstechnik*, ser. Technologien für die intelligente Automation. Springer Vieweg, 2017, pp. 97–106.

The work presented in this chapter has been carried out in the context of the research project "Secure and Semantic Web of Automation" (SeWoA) funded by the Austrian Research Promotion Agency.

4.1 Introduction

In the life cycle of buildings, plenty of different disciplines and companies are involved. Therefore, a unified information base for the whole building and its components and technical equipment is highly beneficial. One approach aiming at this goal is Building Information Modelling (BIM). There exist standards mostly focusing on architectural and

building physical properties, like the Green Building XML (gbXML) schema [78] or the Industry Foundation Classes (IFC) [77]. A further improvement would be to include a description of the technical equipment and building automation components as well, and to make the resulting model accessible in a unified way. This chapter aims at a holistic knowledge base for buildings providing views on static building and automation systems configuration as well as runtime information, i.e., access to live values of datapoint originating in the building automation systems. This includes sensor values, setpoints, calculated or aggregated values as well as time series of historical values.

By the future establishment of the Internet of Things (IoT) [96] in this area with numerous IPv6-based end devices, the degree of automation and the resulting complexity will further raise. Besides highly efficient control and maintenance strategies, use case scenarios like demand side management, load shifting, and energy feedback are future goals of Building Management Systems (BMSs). To this aim, not only building-inherent information sources for energy demand predictions but also external information concerning forecasts for weather conditions and the production of renewable energy sources (e.g., solar or wind) as well as current energy market prices need to be taken into account. For reaching the ambitious goals regarding energy savings and smart consumption of renewable energy within the context of smart grids, a further interoperability problem needs to be solved.

Ontologies, i.e., the patterns behind a knowledge base, suite well for describing buildings and their components. There is already previous work done in this field like [107] and [156]. [157] provides a broad overview on existing ontologies from the domain. Besides the sophisticated modelling capabilities of ontologies, a knowledge base supports also an easily accessible semantic query (SPARQL Protocol and RDF Query Language (SPARQL)) interface which enables access to all building related information relevant for the whole building lifecycle. Application scenarios enabled by this approach are for example predictive maintenance, energy management and optimisation. A future application might be an intelligent control system taking the knowledge base as an information source to found its decisions on control strategies on it.

The integration attempt proposed in this thesis encompasses two main levels: the first level integration, i.e., creating a technology-independent and abstract integration layer unifying access to the plethora of different (legacy) automation systems operating a modern building has already been presented in Chapter 3. This shall further include upcoming IoT devices. OPC Unified Architecture (OPC UA) was utilised for legacy systems integration whereas for IoT devices, Open Building Information Xchange (OBIX) will provide the necessary semantic interfaces. The collected information from the subsystems is concentrated in a Knowledge-Based System (KBS). This is denoted as second level integration. By definition, such a system is in turn set up by two components: an ontology which is a description logic-like model (a schema in the Terminological Box (TBox) of the KBS) of the universe of automation systems, building geometry and physics, user behaviour and external conditions. This ontology mainly builds upon the Think-Home open vocabulary [156] and extends it by a comprehensive device model and also access control mechanisms for each datapoint. The second component of the KBS is

constituted of the actual individuals (the instances or the Assertional Box (ABox) of a KBS) representing the actual building, its environment and the automation equipment. A process for an automated generation of these individuals from OPC UA and OBIX information models, which are written in XML, by means of an Extensive Stylesheet Transformation (XSLT) [34] will be elaborated in Chapter 5.2. Since the OBIX and OPC UA information models can also be automatically instantiated by parsing engineering tool data or by running discovery procedures on building automation networks [158], a fully automated engineering chain throughout the integration layer and the KBS will be created.

As already mentioned, there exists a query language for knowledge bases called SPARQL. Via this interface, semantic queries can be executed. To illustrate the capabilities of SPARQL, examples for possible queries include but are clearly not limited to:

- Are every switching and dimming actuators of a distinct floor in “off” state?
- Which lamps in a building have exceeded a distinct operating time?
- Which lamps in a building are broken?
- How many rooms of a building are occupied?
- Which unoccupied rooms at the southern or western front of the building have their shutters in “down” position?

The SPARQL interface answers with boolean values or with the respective positive instances. On the other hand, it is also possible to update property values in the KBS by SPARQL statements:

- Perform a “central full open” to all shades of a distinct building part.
- Reset all cycle counters exceeding a distinct value.
- Activate or deactivate, respectively, proper electricity consumers for demand side management or flexibility trading purposes.

The changes made in the KBS shall be propagated via the first level integration layer to the underlying automation networks in order to update actual datapoint values.

4.2 Ontology modelling

Information entities in a knowledge base are inherently put in relation to each other in a matter that it correlates to the reality it is intended to describe. It is represented in form of a graph built on a system of subject-predicate-object relations. Subject and objects

are entities whereas predicates constitute the type of relations among them. Additionally, typed entities, and also restrictions on relations like a domain and a range as well as cardinalities are supported. Semantic Web concepts are used to address (HTTP URIs), to structure (Resource Description Framework (RDF) and SPARQL) and to cross-reference (also using HTTP URIs) other resources on the Web. This fulfils the principles defined in the concept of *linked data* [159].

The “SeWoA" ontology, which is developed within this chapter mainly bases on the ThinkHome ontology [156]. ThinkHome in turn combines and extends a number of independent, pre-existent ontologies. For realisation of the previously defined uses cases, two parts of ThinkHome are most relevant, the “architecture and building physics ontology” and the “energy and resources ontology”. The former provides geometrical and structural information about a building, the materials in use and their physical properties. In [156], a transformation scheme from gbXML to this ontology is already described. The defined XSL transformation process encompasses both the model level and the instance level. By this transformation scheme, an automated mapping from the gbXML model to ontology concepts (TBox) as well as building up the according instances (ABox) from gbXML files representing actual architectural projects. The “energy and resources” ontology describes the technical equipment of a building like Heating, Ventilation and Air Conditioning (HVAC) as well as lighting applications, the control systems, i.e., the building automation components based on different technologies including a fine-grained state model to describe the physical state of the devices underlying processes and quantities. Within this chapter, the device model has been extended to expose the most-widespread used field devices in a building like lighting and shading actuators, brightness and temperature sensors, presence detectors and air quality sensors. The device and datapoint semantics, i.e., the underlying physical quantity, engineering units and value ranges, included in the knowledge base are gathered from OBIX and OPC UA information models from the first level integration layer. These models can not only be exposed by integration servers providing unified access to legacy building automation resources. The integration of OPC UA-enabled components or IoT devices natively hosting an OBIX server is also an option. The ontological representations of these devices include meta information about the energy consumption for energy management purposes and also properties for counting operating hours and cycles. By these means, predictive maintenance measures of e.g., lamps, shutters and valves are enabled.

In order to realise a flexible, consistent and fine-grained access control concept it appeared most convenient to implement the necessary mechanisms within the KBS. To this aim, the Social Semantic SPARQL Security for Access Control (Social Semantic SPARQL Security for Access Control (S4AC)) [47] vocabulary has been integrated into the SeWoA ontology and properties representing live values of building automation resources have been enriched with concepts defined in S4AC. The S4AC is based on Access Conditions formulated as SPARQL ASK queries. If such a query evaluates as true, the access condition is said to be verified. The so-called Access Evaluation Context models the relation between resources and requesting users. By Access Tagging Rules the link

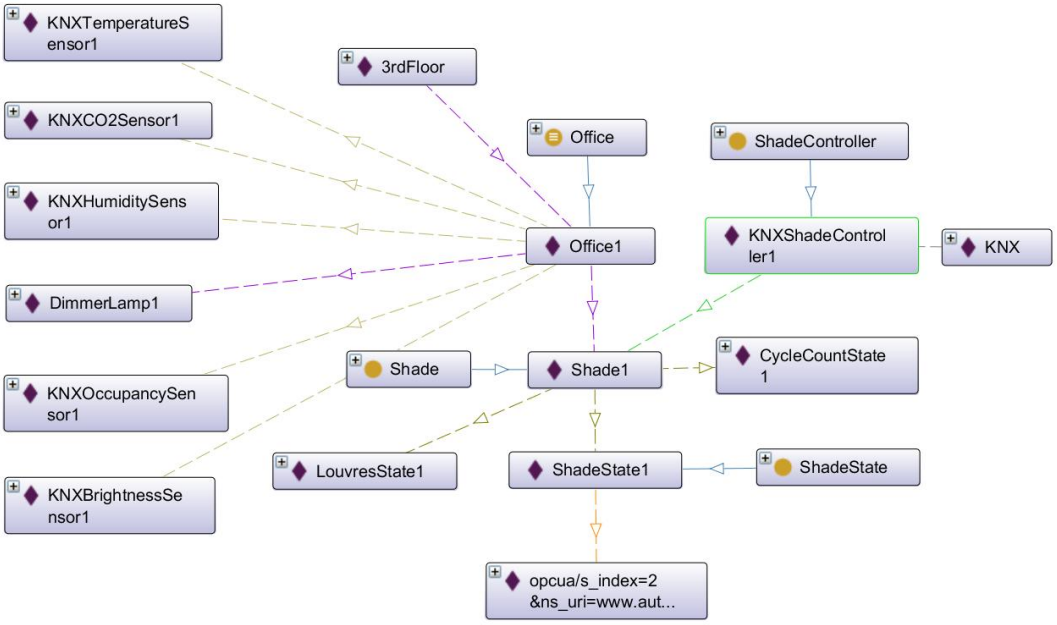

Figure 4.1: SeWoA ontology instance

between Access Conditions, Access Evaluation Contexts and privileges (read, update, delete, create, append) is established. A tag set defines the names of RDF graphs which are concerned by the Access Conditions. Having these concepts integrated into the SeWoA ontology, it is possible to define access policies of arbitrary complexity which scale from single datapoints to any named RDF (sub-)graph.

Figure 4.1 shows a snippet of an Web Ontology Language (OWL) model of an office room equipped with a number of room automation components. OWL concepts are denoted by yellowish circle symbols, whereas OWL individuals are illustrated by purple diamonds inside the boxes. The `Office1` individual is an instance of the Office concept. This office room shown here is part of the `3rdFloor` which in turn is embedded in a larger building context. For simplicity reasons, the overlaying hierarchy is not shown in this figure. The individuals representing sensors for temperature, air quality, occupancy and brightness (`KNXTempertureSensor1`, `KNXCO2Sensor1`, `KNXHumiditySensor1`, `DimmerLamp1`, `KNXOccupancySensor1`, `KNXBrightnessSensor1`) are associated with the `Office1` individual by `hasSensor` object properties. In order to differentiate the semantics, devices like the dimmer lamp and the shade are referenced by `isIn` object properties. The `Shade1` individual and its object properties are expanded for clearance in this figure. On one hand, the shade is linked to a `KNXShadeController` instance which in turn has KNX as a building automation network type. The physical state of the `Shade1` is represented by `LouvresState1` and the `ShadeState1` individuals. For maintenance purposes, a cycle counter has been added, too. The actual value

of the distinct states is encapsulated in individuals of the OWL class `State-Value` like the entity at the very bottom of Figure 4.1. In a first version of the SeWoA ontology, which was published in [160], the URIs of the individuals of these `State-Value` classes encode all the parameters necessary to uniquely identify a node (in this case, a variable node) in the address space of a first level integration resource. By this measure, the logical link between the KBS instances representing actual datapoints and the underlying values in the integration technology (OPC UA, OBIX or other Web service protocols) is established. Address encodings defined here encompass paths to OPC UA variable nodes, contracts on an OBIX server or Constrained Application Protocol (CoAP) resource locators to devices that are natively addressable by restful Web services (WSs). Via this referencing mechanism, the datapoint individuals in the knowledge base have a link to physical networks and components in a building's installation which a KBS can process. SPARQL queries and updates on the knowledge base can hereby deliver live values from the underlying field devices and automation components and allow to influence setpoints or every other writable variable.

The reference designation system defined in IEC 81346 is used in a further development of the ontology, published in [161]. Hereby, convergence is achieved in both static information model domains: BIM and automation systems configuration. This is essential for the following automated knowledge base instantiation which is described by means of the following example.

We consider a temperature sensor located in room number 22 of the fourth floor of building DE. The resulting reference designation of this sensor is `-BT04+DE+04+22` where `-BT04` describes the product aspect of being a temperature sensor and `+DE+04+22` the location aspect. Provided that in the devices' OBIX contract as well as in the gbXML model the same building topology naming scheme is used, the temperature sensor can be automatically associated to the respective room, floor and building. This principle is illustrated in the ontology snippet shown in Figure 4.2 where the yellowish circles denote OWL classes and the purple diamonds indicate OWL instances. The instances are organised in a hierarchical way with a building individual on the top. The fourth floor of this building contains a room labelled DE0422. The temperature sensor instance labeled `-BT04+DE+04+22` is assigned to this room via a `IsIn` reference. A textual description of the device instance (e.g., "Room Temperature Sensor") can be given as an OWL annotation.

Like already introduced in initial version of the SeWoA ontology, the goal of this knowledge base concept for building management is not only to consolidate static building information but also runtime data from the automation systems deployed in a building. In Figure 4.2 the `-BT04+DE+04+22` temperature sensor exposes a state `TemperatureState4` and a value `TemperatureStateValue4` (an ontology design concept defined in [162]). Analogously, the lighting `-EA02+DE+04+22` has a state `OnOffState02` and a value `OnOffStateValue02`. The state value individuals are kept up to date with the actual sensor or actuator values during runtime.

The logical connection between the state value representation in the knowledge base and

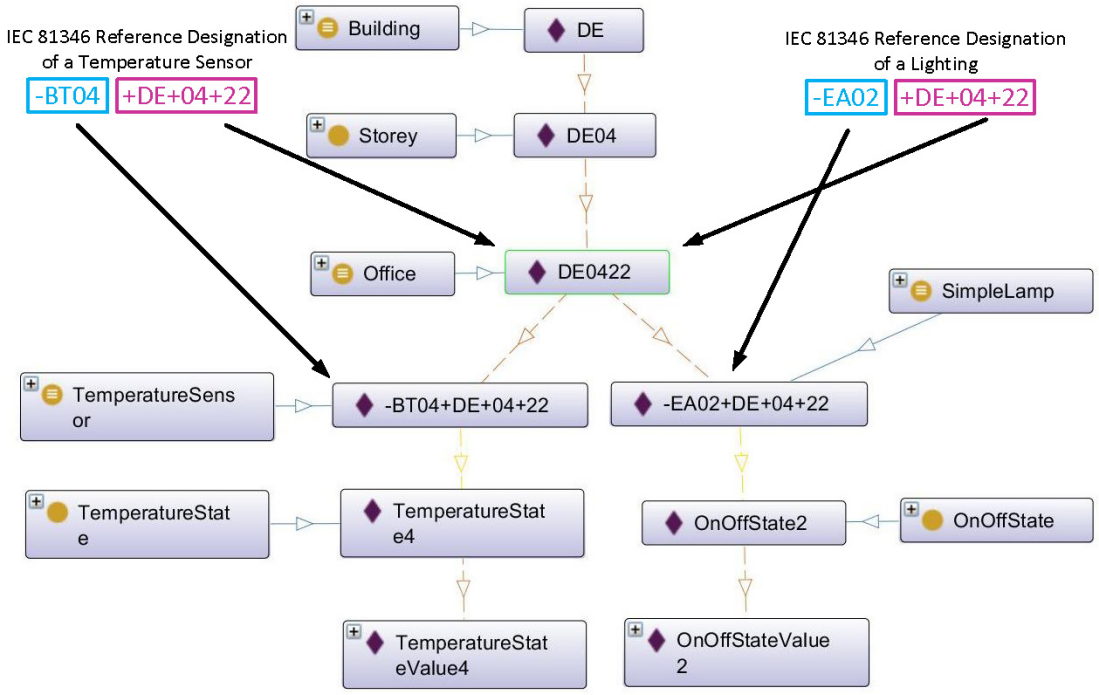

Figure 4.2: Building and device ontology instance aligned by IEC 81346 reference designations

the actual datapoint in the underlying automation system is in the extended version of the ontology realised by an annotation with the corresponding datapoint address. In detail, a data property carrying the the datapoint address is referenced by the state value individuals. This alteration has been done for flexibility reasons to decouple the datapoint address, which might change sometime from the state values' URI. In the following example, this concept is shown for the already introduced temperature sensor and the lighting `-EA02+DE+04+22` located in the same room. In order to demonstrate the mapping from two integration standards used in the building automation domain, this process is illustrated by means of the temperature sensor assumed to have an OBIX interface and the lighting being integrated via OPC UA. At the moment, there do not exist general mapping schemes from OBIX contracts and OPC UA object types to the OWL knowledge base representation because of the high diversity in automation components where each type needs to be treated individually. Nevertheless, for a number of further field devices (among others, a presence detector, an air quality meter and a sunblind actuator), a mapping from OBIX and OPC UA to OWL has been defined.

The following listing shows an OBIX object of a temperature sensor:

```
<obj name="-BT04+DE+04+22" href="/hvac/Sensors/TemperatureSensor04"
        displayName="RoomTemperature">
```

```
 <real name="value" href="value" val="24.23"
       unit="obix:units/celsius"/>
</obj>
```

It contains the name of the object, the destination of the resource (`href`) and a human readable display name. Besides the current measurement value, also the engineering unit is provided. The corresponding knowledge base individual in RDF/XML notation representing a state value (i.e., a datapoint object) looks like the following:

```
<owl:NamedIndividual rdf:about="&SeWoA;TemperatureStateValue4">
 <rdf:type rdf:resource="&SeWoA;TemperatureStateValue"/>
 <SeWoA:realStateValue rdf:datatype="&xsd;float">24.23f
       </SeWoA:realStateValue>
 <SeWoA:hasOBIXAddress rdf:datatype="&xsd;string">
       http://128.130.56.51:8080/hvac/Sensors/
       TemperatureSensor04</SeWoA:hasOBIXAddress>
 <SeWoA:hasNativeUnit rdf:datatype="&xsd;unitEnum">Celsius
       </SeWoA:hasNativeUnit>
</owl:NamedIndividual>
```

Like shown in Figure 4.2, this `TemperatureStateValue4` individual is embedded in the context of another individual (`-BT04+DE+04+22`) standing for the physical temperature sensor. The state value individual shown in the above listing provides, besides type information, all the attributes exposed by the OBIX object. The present datapoint value is exposed by the `SeWoA:realStateValue` element. Via the `SeWoA:hasOBIXAddress` element containing the URL of the OBIX object, the link from to the knowledge base individual to the associated live OBIX datapoint is realised.

Like for OBIX, the OPC UA specification defines an XML Schema Definition (XSD) for an XML format in which OPC UA information models can be saved. Usually, OPC UA modelling tools export information models in XML files of this format. OPC UA servers can load such XML files and instantiate their address spaces which are then exposed to OPC UA clients accordingly. In the following, an excerpt of such an XML representation describing an OPC UA variable node is shown. Variable nodes are used in OPC UA to model datapoints. Besides a `BrowseName` attribute, which is set to the reference designation of the lighting `-EA02+DE+04+22`, there is also a human-readable localised DisplayName attribute reading `LightOnOffState`. The actual value is of type `Boolean` and currently set to `false`.

```
<Node i:type="VariableNode">
 <NodeId>
  <Identifier>ns=3;i=325433</Identifier>
 </NodeId>
 <NodeClass>Variable_2</NodeClass>
 <BrowseName>
  <NamespaceIndex>5</NamespaceIndex>
  <Name>-EA02+DE+04+2</Name>
 </BrowseName>
 <DisplayName>
  <Locale>en</Locale>
```

```
   <Text>LightOnOffState</Text>
 </DisplayName>
...
 <Value>
  <Value>
   <Boolean>False</Boolean>
  </Value>
 </Value>
...
</Node>
```

The XML fragment describing the OWL individual resulting from the previous OPC UA variable node is shown in the following:

```
<owl:NamedIndividual rdf:about="&SeWoA;OnOffStateValue2">
 <rdf:type rdf:resource="&SeWoA;OnOffStateValue"/>
 <SeWoA:realStateValue rdf:datatype="&SeWoA;OnOffStateValueEnum">Off
        </SeWoA:realStateValue>
 <SeWoA:hasOPCUAAddress rdf:datatype="&SeWoA;OPCUAAddress">ServerURL=
        "opc://localhost:4840" NamespaceURI=www.auto.tuwien.ac.at
        IdentifierType= Numeric Identifier=325433
        </SeWoA:hasOPCUAAddress>
</owl:NamedIndividual>
```

Analogously to its temperature sensor datapoint sibling, this `OnOffStateValue2` datapoint individual is also referenced by a device individual (`-EA02+DE+04+22`) representing the actual lighting. Besides the type information and the actual state value, there is an XML element containing the address information necessary to access the associated OPC UA variable node. It can be seen as a container for a network-wide node address consisting of the server URL, and the OPC UA NodeId consisting of the `NamespaceURI`, the `IdentifierType` and the numeric `Identifier` itself.

4.3 A Semantic Web integration platform

4.3.1 Architecture

For a prototypical implementation of the knowledge base, the Java based open source Semantic Web framework Apache Jena[1] has been chosen as a basis. The prototype's architecture, like already mentioned, can be seen on top of the automation pyramid in Figure 4.3. As a test environment for this prototype, Zolertia[2] IoT devices as well as an OBIX server running the integration middleware IoTSys[3] has been used. The Apache Jena framework provides an RDF triple store, which is instantiated by means of the previously generated XML files, the output of the XSLT. It also comes with a SPARQL endpoint which allows executing semantic queries and updates on the RDF triple store. The RDF triple store holds a knowledge representation of the building topology and the automation systems configuration. Additionally to this static information, it shall also incorporate

[1] https://jena.apache.org/
[2] http://www.zolertia.com
[3] https://github.com/mjung85/iotsys

a live image of the underlying automation systems datapoints. This is achieved by an OPC UA as well as an OBIX client module which communicate with the triple store via an Application Program Interface (API). Via their network interfaces they establish connections to the underlying OBIX and OPC UA resources. More precisely, OPC UA and OBIX client modules register subscriptions on underlying datapoints corresponding to the knowledge base individuals instantiated in the triple store. The data properties in the RDF triple store and the underlying datapoints are mutually synchronised, either via the integration servers (OBIX and OPC UA) or by directly interacting with IoT-enabled devices. Figure 4.4 illustrates the interaction of the semantic BMS with an underlying OBIX resource, i.e., data synchronisation in upwards and downwards direction. The OBIX or OPC UA addresses necessary to access the correct resources are gained from the `hasOBIXAddress` or from the `hasOPCUAAddress` properties of the respective OWL individuals. After an initialisation phase, where the building knowledge base creates an OBIX watch object on the OBIX server, it registers all OBIX objects corresponding to OWL individuals for change-of-value observation by calling a `WatchIn` operation. In this example, this is only shown for the OBIX `TemperatureSensor04` object corresponding to the `-BT04+DE+04+22` OWL individual. The OBIX device acknowledges the `WatchIn` request by a `WatchOut` object containing the current value of the respective OBIX object. From now on, the building knowledge base continuously polls for value changes. If this is the case, the OBIX client gets informed and updates the value of the `hasNativeValue` data property of the datapoint individual via the API of the triple store. Value buffering achieved this way is necessary for performance reasons with respect to SPARQL queries traversing a huge number of datapoint individuals for their value. Otherwise, if the current values would have to be requested from the OBIX and OPC UA resources when the query is issued, a considerable load in the underlying building automation network would be the result. This would in these cases lead to an unreasonable response time of the query.

This SPARQL query delivers all simple (switched) lamps which exceeded an operating hours count of 1000h:

```
SELECT  ?lamp ?hours
WHERE {
        ?lamp rdf:type sewoa:SimpleLamp.
        ?lamp sewoa:hasOperatingHours ?operatinghours.
        ?operatinghours time:hours ?hours
        FILTER(?hours > 1000)
        }
```

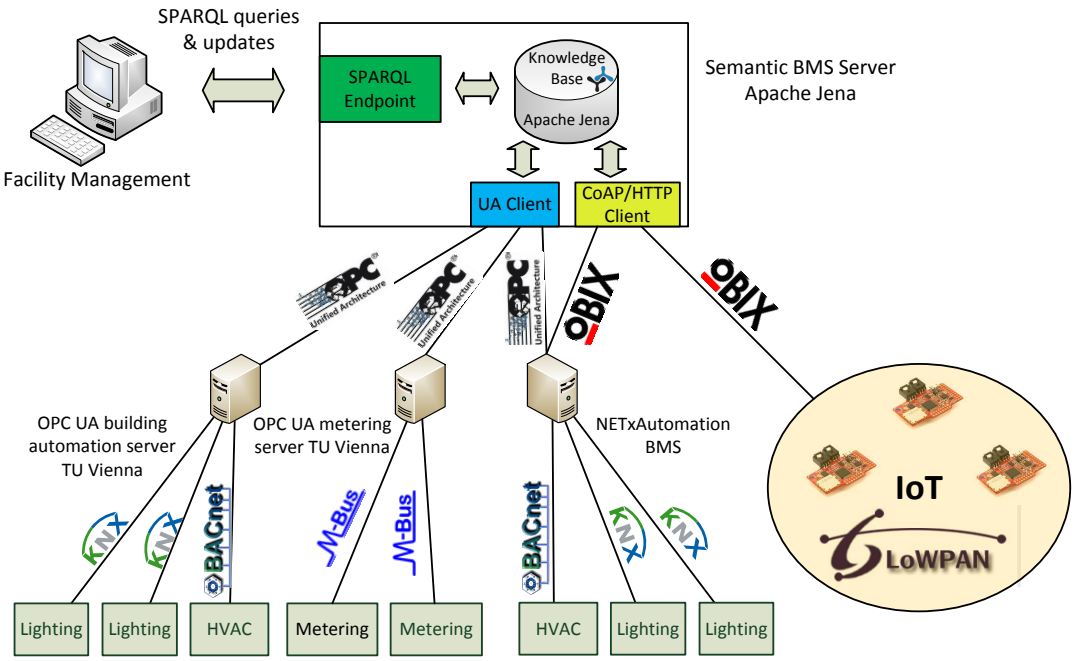

Figure 4.3: Knowledge-based BMS architecture

The following update statement (in SPARQL, this is done via DELETE and INSERT requests) causes a "central fully open" to all shades of a building:

```
DELETE { ?value sewoa:realStateValue ?datavalue }
INSERT { ?value sewoa:realStateValue "100" }
WHERE {
        ?shade rdf:type sewoa:Shade.
        ?shade sewoa:hasState ?state.
        ?state rdf:type sewoa:ShadeState.
        ?state sewoa:hasStateValue ?value.
        ?value sewoa:realStateValue ?datavalue.
        }
```

4.3.2 Runtime Interaction

In Figure 4.4, two exemplary SPARQL queries are shown. The first one requests for all temperature sensors with a reading greater than 20.0 degrees. The query returns the `-BT04+DE+04+22` individual and the buffered value of its associated data property. It is also possible to update states of the automation components by means of SPARQL queries which is shown in the following query in the same figure. The meaning of this query is to perform a central-off to all lights in the fourth floor `DE04`. Such a write access is realised using the SPARQL Update language by first removing the respective data properties by a DELETE statement followed by an INSERT statement which creates new data properties with the new `false` value. For every updated individual, the event

listener of the RDF triple store API triggers the OBIX client to send write requests containing the new value to the regarding OBIX objects on the underlying resource.

4.4 Results

In this chapter, an approach for second level integration of BIM and automation technology data into a BMS has been shown. A concept of a holistic knowledge base encompassing building information, automation systems and live process values was introduced. Moreover, a workflow of aligning information originating from different sources, namely topological building models (BIM) and automation systems OBIX or OPC UA interfaces using the IEC 81346 reference designation system has been shown. This information transformation process has been realised by means of an XSLT which has been empirically evaluated for a limited set of entities. In order to show runtime interworking of this automatically instantiated knowledge base with actual automation components, an Apache Jena-based prototype equipped with OBIX and OPC UA client functionality has been implemented. An open source project named OpenKB4BMS[4] including both the transformation stylesheets and the knowledge base implementation has been published on GitHub. It can already be concluded that a ontology-based KBS in combination with a pervasive information flow down to the field level of automation systems and building information allows the realisation of use cases that would not be possible with a conventional approach. The plus of the concept introduced is the high flexibility provided by a SPARQL interface.

[4] https://github.com/afernbach/openKB4BMS

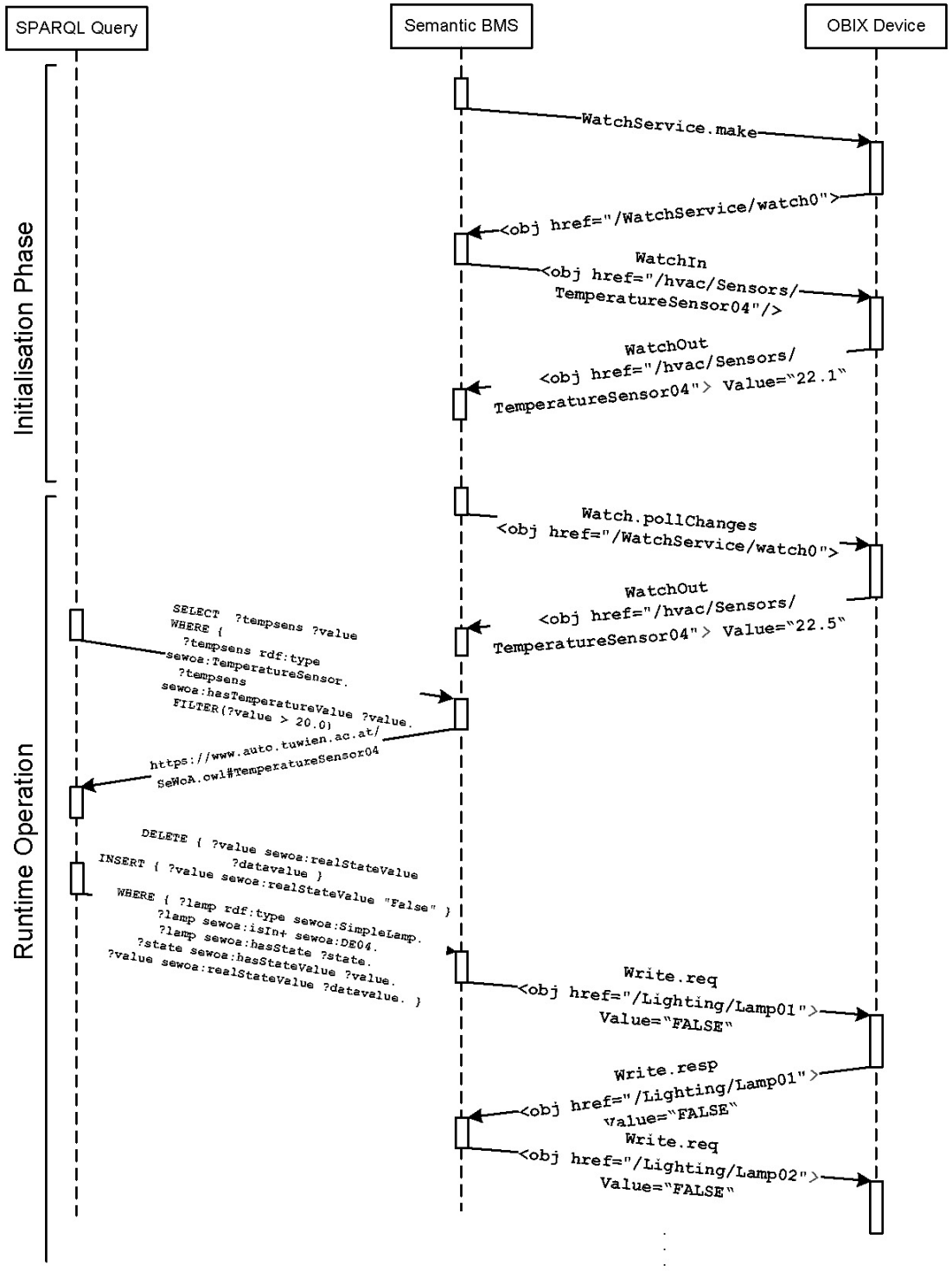

Figure 4.4: Runtime interaction of a building knowledge base with an OBIX device

CHAPTER 5

Building automation systems and security engineering

This chapter is based on the contributions of the author to the following publication:

A. Fernbach and W. Kastner, "Semi-automated engineering in building automation systems and management integration," *26th IEEE International Symposium on Industrial Electronics (ISIE)*, Jun. 2017.

A. Fernbach and W. Kastner, "Certificate Management in OPC UA Applications: An Evaluation of different Trust Models," *Proc. of 17th IEEE Conference on Emerging Technologies and Factory Automation (ETFA '12)*, Sep. 2012.

The work presented in this chapter has been carried out in the context of the research projects "Web-based Communication in Automation" (WebCom), "Secure and Semantic Web of Automation" (SeWoA) and "Kognitive Regelstrategieoptimierung zur Energieeffizienzsteigerung in Gebäuden (KORE)" funded by the Austrian Research Promotion Agency.

5.1 Introduction

Building planning as well as automation systems engineering involve numerous disciplines participating in a many-phase process. However, the information transitions between these different phases are mostly carried out manually which results in multiple sources of errors. Additionally, information which is already present is repeatedly manually re-entered during the whole workflow. This especially applies for Building Automation System (BAS) engineering where on the other hand the majority of necessary information

is already generated during the planning phase. To address this issue, the first section of this chapter proposes a workflow of semi-automatable engineering for BASs, and also first level and second level integration. This workflow is presented by means case studies on representative technologies from the respective levels.

In automation systems integration, where networks are interlinked to the office and IT infrastructure and even opened to the World Wide Web, strong security measures are required. OPC Unified Architecture (OPC UA) provides a powerful and inherent security model. The same applies for HTTPS-based [163] Web service communication of Open Building Information Xchange (OBIX) servers and SPARQL Protocol and RDF Query Language (SPARQL) endpoints. These mechanisms in both technologies rely on software certificates. In an integrated automation system, where OPC UA and Web services (WSs) are applied, also a strategy must be defined how to manage these certificates, i.e., an organised way of distribution, validation and revocation needs to be found. In general, there exist different concepts of how to achieve this goal. Moreover, there are various, in some cases platform dependent frameworks available which assist the developer in implementing a suitable concept. To this aim, the second part of this chapter gives an overview of these concepts and frameworks and discusses their positive and negative aspects depending on the structure of different environments in which OPC UA and Web service applications shall be embedded.

5.2 Semi-automated engineering based on planning data

Today's lifecycle of a building normally starts with following consecutive phases involving a considerable number of different disciplines:

- Strategic planning
- Preliminary studies
- Project planning
- Construction bidding
- Realisation
- Operation, including maintenance and retrofitting

The information flow between these different phases is usually not carried out in a consistent way leading to information loss and misinterpretations finally resulting in severe problems and delays during the realisation phase. Initial requirements to the building physics and building services are fulfilled incompletely and thus construction and integrator companies are confronted with long lists of deficiencies.

In this context, one of the most affected trade is the integrators of technical building equipment. The information flow from building services planning to automation systems engineering involves extensive manual information transfer processes. Automation systems planning data needs to be interpreted by the engineers and technology specific engineering tool projects for device and system configuration need to be set up accordingly. These tasks involve repetitive and tedious steps which are error prone and time consuming. Additionally, many planning offices and integrator companies use their individual designation systems for automation systems and components which often provide only low expressiveness and therefore involve the danger of ambiguous systems description. This is again an error source during planning and engineering. It also exacerbates vendor independence and later third-party integration of example for data monitoring applications.

The goal of this section is to describe a process of continuous and machine readable information flow from building services planning to automation system engineering and also management gateway integration. Planning information shall be - as far as possible - entered once and reused during the whole workflow.

There have already been published concepts addressing this question. The most elaborated concepts in this field are [126] and [164]. They describe a multi-tiered method of automated design. Main attention is turned on fieldbus and automation systems but these approaches have a weak focus on the integration level. There exists also a commercial tool providing a data exchange interface between CAD programs and the KNX engineering tool Engineering Tool Software (ETS). This so-called "ETS APP PROJECTDATAEXCHANGE" from IT Gesellschaft für Informationstechnik mbH[1] also by its nature focuses on the field and automation level.

The general information flow of planning a building and its automation system components to the engineering at automation level and integration level is shown in Figure 5.1. The orange-colored arrows indicate the application of models, i.e., the patterns behind the instances generated during this workflow. Models in this application field can be Building Information Modellings (BIMs), standards defining designation systems for buildings (like EN ISO 4157 [165]), technical equipment (IEC 81346-1 [79]) or technology-specific application models like for KNX or OPC UA. During the planning and engineering process of a building, the actual problem is formulated and instantiated applying these models at multiple states of the overall progress.

In the beginning, when the building owner, architects, civil engineers and building automation engineers carry out the planning phase the use of tools supporting BIM standards like Green Building XML (gbXML) and designation systems like EN ISO 4157 and IEC 81346 reduces error sources and improves the maintainability and reliability of produced planning data. Examples for such tools are the widespread products *AutoCAD Architecture* and *Eplan Electric*. The resulting planning data, which can be seen as a *holistic building information model*, includes structural and geometrical information of

[1] http://www.it-gmbh.de/

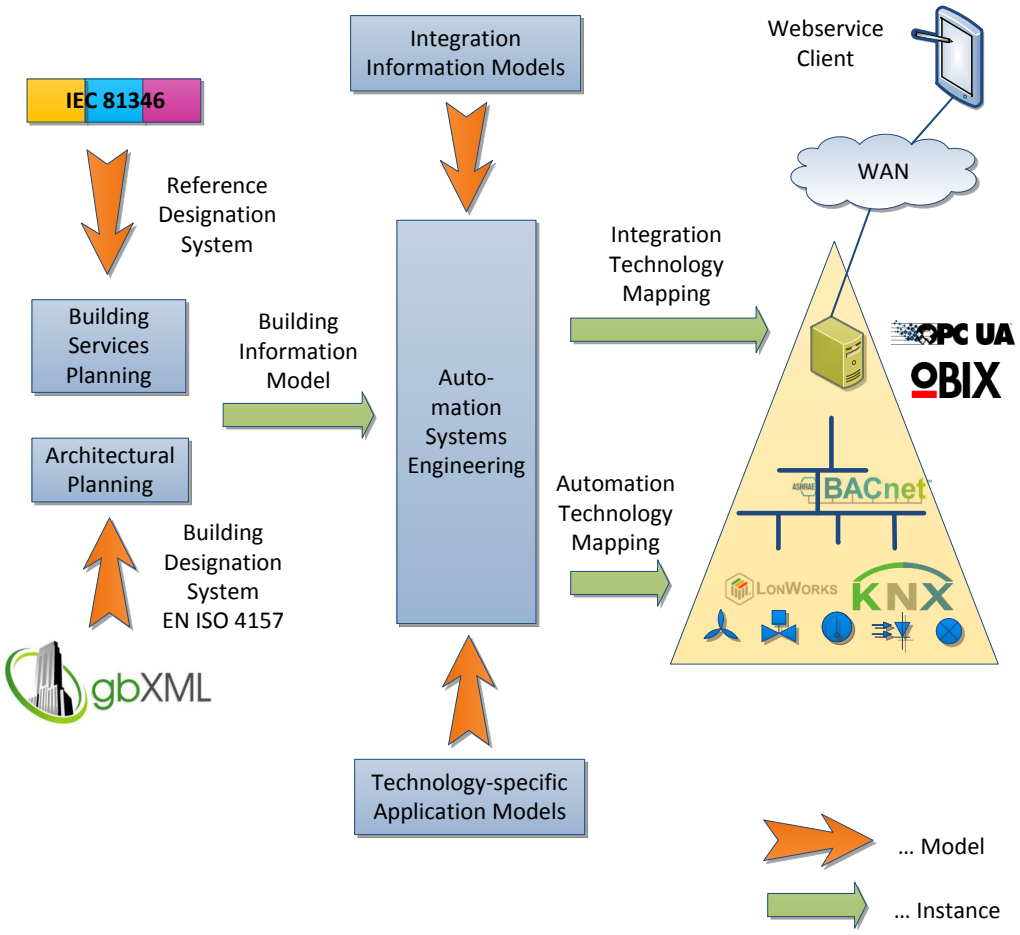

Figure 5.1: Engineering workflow

the building as well as structured information about the technical equipment. The goal is to include as much as possible of the information needed in following engineering phases already at this point. This is the prerequisite for implementing (semi-)automated technology mapping processes when it comes to automation systems engineering.

The pyramid-shaped object on the right side of Figure 5.1 shows the typical structure of building automation systems. As already familiar, field devices based on KNX or LonWorks interact with the physical processes in the building. BACnet [11] devices are usually used at automation and management level. Multi-protocol integration and gateway interaction is also located at the management level on top of this pyramid model.

The planning information previously generated (i.e., the holistic information model) constitutes the information source for the next step, namely for mapping abstract

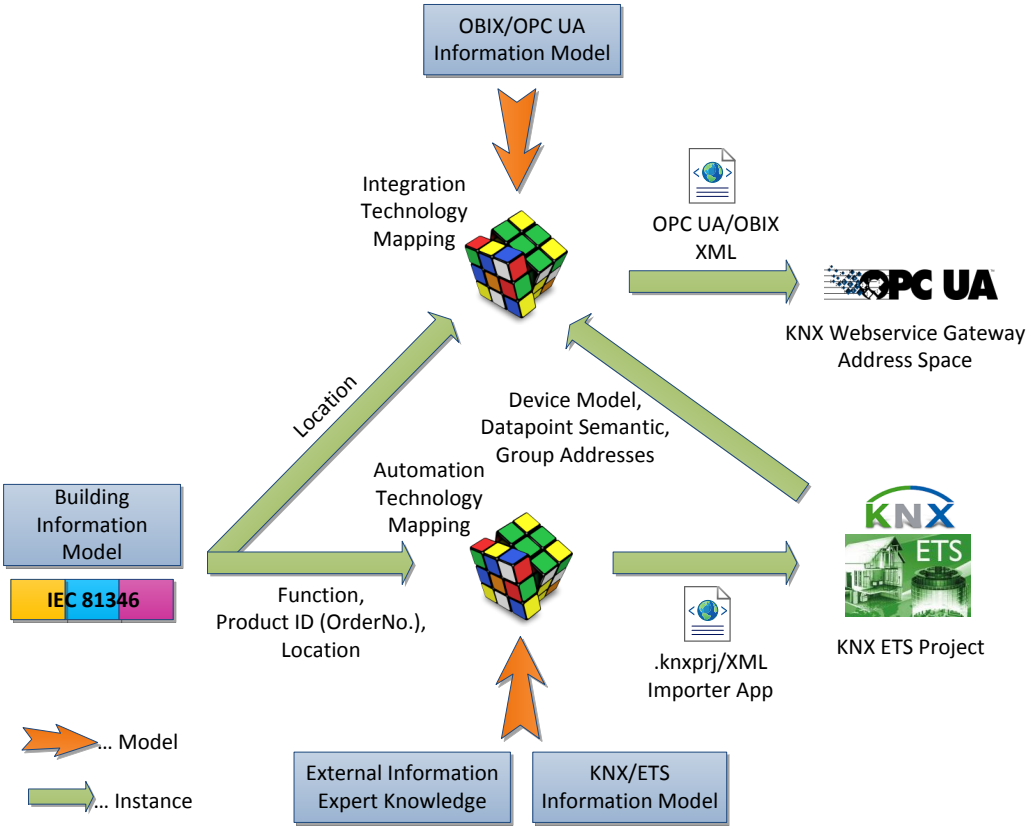

Figure 5.2: Automated engineering

building services to specific application models of building automation technologies. This process is normally understood as automation systems engineering. Planning data is manually processed and put into technology-specific and in most cases (except KNX which uses the ETS) vendor-specific engineering tools. This process is carried out on multiple layers of automation systems. Initially for field and automation technologies but later or even in parallel when it comes to Supervisory Control and Data Acquisition (SCADA) or Building Management System (BMS) integration or middleware engineering on top level of automation systems. However, parts of this process can be automated utilising information from properly structured planning data.

Figure 5.2 shows this technology mapping process in detail. It takes planning information consisting of structural building information and automation systems description using the IEC 81346 reference designation system. Hereby, following aspects are included regarding automation equipment:

- Location
- Function

- Product ID (order number)

The location aspect shares a common designation system with the building information model and, therefore, a unique assignment of automation equipment to a topological building element is assured. The KNX ETS technology mapping (engineering) relies on the product aspect and the location aspect, whereas the functional aspect is optional. In an ETS project, KNX devices are uniquely identified by their order number. This allows to add KNX devices according to the list of IEC 81346 reference designations to an ETS project in a straight forward manner. In a next step, the building view in the ETS is instantiated with building topology defined in the BIM. Following the shared building designation system, KNX devices are assigned to their intended building topology elements. This results in an ETS project skeleton which can be automatically generated. The ETS App Application Program Interface (API) allows to interact with the ETS via third-party applications providing this information to the ETS. Still, there are parts of the engineering process that need to be carried out manually. On one hand, proper configuration of devices by setting their parameter requires information which is not available by the proposed planning data. Moreover, the definition of functional relationships on the level of datapoints which is achieved by the definition of communication groups is not automatable using this concept since the designation systems in use do not support behavioural modelling. AutomationML[2] might be considered to fill this gap. This group communication engineering is therefore a remaining manual step which is in practical ETS project engineering carried out by assigning shared group addresses to the intended communication partners.

In order to instantiate a first level integration middleware based on, e.g., OPC UA, additional information is needed from the KNX ETS project. The planning information does not include a device model giving a description of the set of datapoints a device exposes. Also the datapoints' semantics is missing, i.e., the datatype, engineering unit and value range. However, this information is contained in the device model of the ETS. The datapoint types where the ETS device model refers to are exactly defined in the KNX System Specification. They include the desired meta information like datatype, encoding, unit and range. In Figure 5.2, this information flow is illustrated by the upwards-heading arrow from the ETS symbol to the integration technology mapping symbol. Additionally to that, also the group address scheme assigned to a KNX installation needs to transferred to such a gateway. Like a native KNX device, it needs the address information to be able to properly access KNX datapoints in the network.

Hereby, the necessary information to instantiate an OPC UA-based middleware is available. The topological structure can be directly gained from the BIM which is in a further step linked to the device-related information. The result can be formalized following the standardised OPC UA XML Schema Definition (XSD) and loaded as an XML configuration file into the middleware implementation.

[2] AutomationML association, https://www.automationml.org

This engineering process concept is illustrated in the following by an example. A model of a building is assumed to have four floors whereas the fourth floor contains two rooms, an office room and a meeting room. This topological elements are labelled using a designation system like the following: Building `DE`, fourth floor `04`, office `23` and meeting room `22`. This results in the unique designations `+DE+04+23` for the office and `+DE+04+22` for the meeting room, respectively.

The gbXML code snippet for the office room is:

```
<Space id="+DE+04+23" conditionType="HeatedAndCooled">
        <Name>Office</Name>
                ...
```

and for the meeting room:

```
<Space id="+DE+04+22" conditionType="HeatedAndCooled">
        <Name>Meeting Room</Name>
        ...
```

The office is equipped with a switching actuator assigned with the IEC 81346 reference designation:

```
=EA=KF03-5WG1562-2AB31+DE+04+23
```

The functional aspect of this device is declared with the hierarchical descriptor `=EA=KF03` standing for a controller device being part of a lighting system. This is followed by the product aspect `-5WG1562-2AB31` representing the ETS order number of the device. `+DE+04+23` denotes the local aspect which conforms to the label of the room where the switching actuator is located. Analogously to that, a temperature controller is denoted with:

```
=EP=KF01-5WG1253-2AB_3+DE+04+22
```

where `=EP=KF01` indicates the functionality of a controller device as part of a heating system. `-5WG1253-2AB_3` stands for the order number of the device and `+DE+04+22` for the local aspect.

5.2.1 Engineering of a KNX ETS project

Following the workflow concept described in the previous section, the information from the exemplary building model and from the device designations is taken - with respect to the possible degree - to generate an ETS project skeleton. The screenshot in Figure 5.3 shows a snippet of the resulting ETS project configuration. The ETS *Buildings* view exposes the hierarchical building topology of the building `+DE` with the fourth floor `+DE+04` and the two rooms `+DE+04+22` and `+DE+04+23`. The strings representing the human readable names of the topological objects (`4th Floor`, `Meeting Room`, `Office`) are taken from the gbXML building model. Beyond these two room elements, the respective KNX devices are instantiated. This assignment is achieved by the order number given by the device designations.

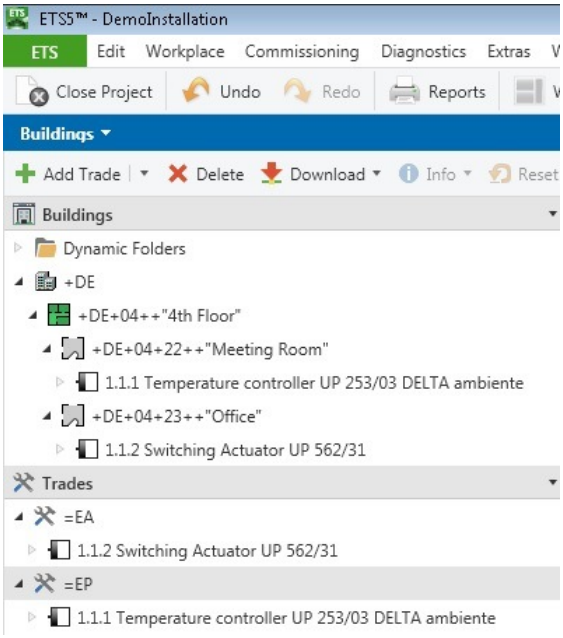

Figure 5.3: ETS project

Additionally to the Buildings view, the ETS provides a perspective called *Trades* where devices can be arranged according to their functionality. This view is used to reflect the functional aspect denoted by the devices' reference designation system. Hence, two trades are instantiated, `=EA` for lighting and `=EP` for heating. Then, the exemplary KNX devices are assigned to these items.

5.2.2 Engineering of first level integration

The following phase of automation systems engineering aims at instantiating a first level integration middleware providing Web service access to the underlying installation. The focus lies on integration by means of OPC UA using the upcoming KNX Web service specification, where the author contributed. Therefore, it differs a little from the previously introduced KNX OPC UA information models. In Section 6.1.2, an overview of this OPC UA representation will be given. Like for the ETS engineering process, already existing information is processed and used to create OPC UA instances. This also includes information derived from the previously generated ETS project. The resulting OPC UA instance which is compliant to the KNX WS specification is shown in Figure 5.4. The `Root` object is the entry point for an OPC UA client exploring the address space of a server. A real OPC UA server exposes a number of folders (containing e.g., ObjectTypes, VariableTypes, ReferenceTypes) at this point which are not shown in this figure except for the `Objects` folder. The set of instance nodes of a server are grouped beyond this item. The orange-colored hierarchy of `ViewType` objects reflects the building topology. It terminates with the two example rooms `DE+04+23++Office`

and DE+04+22++Meeting Room. Via Organizes ReferenceTypes, two Objects representing KNX devices KNXTemperatureController and KNXSwitchingActuator are linked. These two device objects can be additionally browsed via the DeviceSet object which is required by the OPC UA for Devices (DI) companion standard. The KNX WS specification includes this device model as already mentioned. For simplicity, only the switching actuator device shows part of its address space. The ParameterSet object exposes a binary datapoint variable SwitchingValue which is enriched with datapoint semantics about the encoding (TrueState and FalseState properties) and the originating KNX datapoint type (KNXDatapointType). As examples for static device information, three properties informing about the serial number, revision and manufacturer are displayed. The information necessary for instantiating the devices, their static properties and the datapoints originates from the ETS project. One aspect which is not visible in the OPC UA representation are the group addresses which are required for runtime interaction between the KNX WS gateway and the KNX network. They are also taken from the ETS project and saved internally in the gateway.

5.2.3 Engineering of second level integration

When this semi-automated engineering concept is extended to second level integration, the next step is to automatically generate the Knowledge-Based System (KBS)'s Terminological Box (TBox) and Assertional Box (ABox). Therefore, the origins of relevant information must be identified first. In order to create a holistic representation on this level, information from two domains need to be fed into the knowledge base and put into relation, the domain of BIM and the domain of building automation systems.

The BIM domain delivers the structural and topological information about a building, its orientation and its physical properties. A method how to transform a gbXML file into a Web Ontology Language (OWL) model is already treated by [162]. Therefore, the focus of this section lies on defining a process to transfer information regarding automation components and devices into the KBS. Information is organised by three different levels, the meta model level, the models level and the instances level, which are shown in Figure 5.5. On the bottom level, the meta models of OPC UA and OBIX on the source side and the OWL schema on the target side are situated. For all three standards, XSD interpretations exist. The XML schemas of the integration technologies, OPC UA and OBIX define basic datatypes, variable types and object types. So to speak, the fundamental building blocks of OPC UA and OBIX information model are constituted by these meta models. The OWL XML schema describes all available modelling entities like Classes, ObjectProperties, Datatypes, DataProperties and basic OWL axioms. Since the XSDs are defined and published by standardisation bodies (OPC Foundation, Organization for the Advancement of Structured Information Standards (OASIS) and World Wide Web Consortium (W3C)) and therefore they can be seen static, no actual transformation takes place at this level.

The Models tier on the mid level of this hierarchy encompasses - speaking for OBIX and OPC UA - technology specific information models like KNX. These information

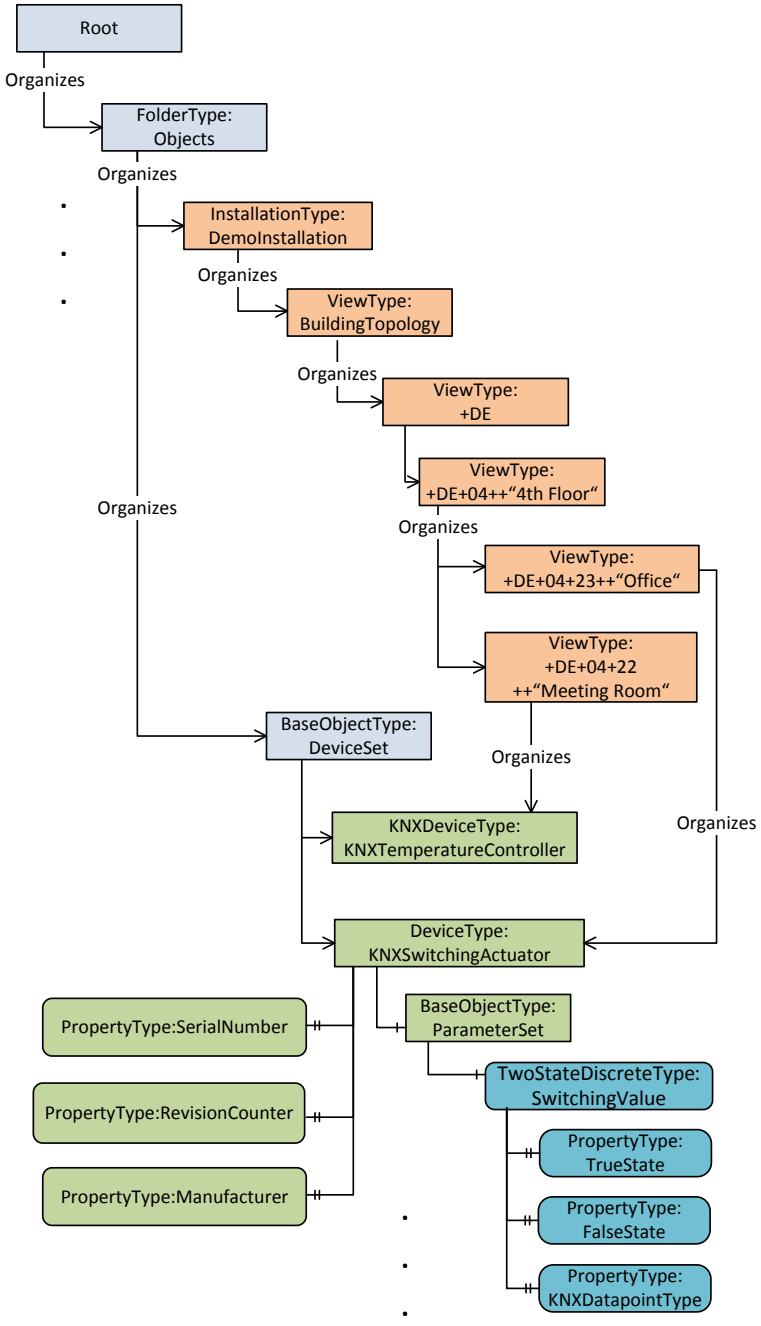

Figure 5.4: OPC UA instance of KNX installation

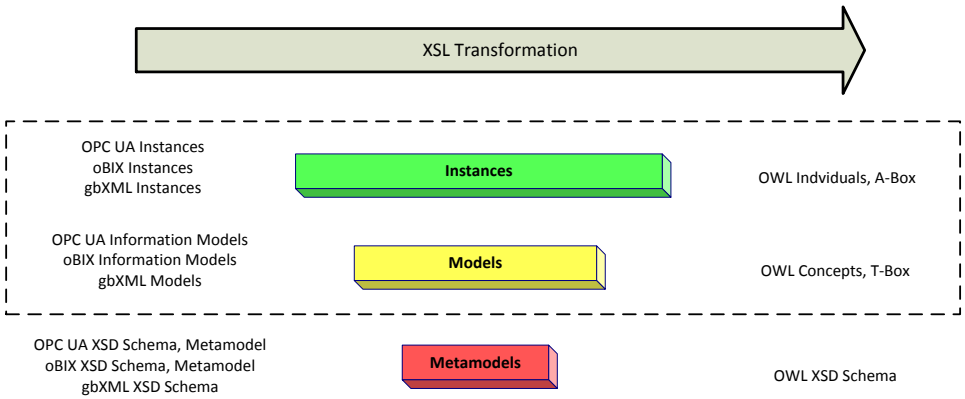

Figure 5.5: Model hierarchy

models describe how distinct devices like a dimming actuator or a shade controller expose their datapoints. On OWL side, the corresponding concepts or classes including ObjectProperties and DataProperties are defined. These modelling constructs set up the KBS's TBox. So, the Models tier contains patterns to be instantiated when it comes to a concrete building automation installation. This applies for both integration technologies and OWL. Transformation rules were defined to automatically generate the TBox of a KBS from OBIX and OPC UA information models. The Instances level is situated on the top of this model hierarchy. Regarding integration technologies, OPC UA and OBIX representations of concrete automation system devices and live process data - i.e., instances of the previously defined information models - are located here. In practice, these instances are exposed by OPC UA or OBIX servers' address spaces and accessible by according clients. Analogously, like illustrated on the right (KBS) side of this figure, individuals derived from OWL concepts which are defined in the underlying Models tier are instantiated here. They form the ABox of a KBS. The ABox hereby represents the automation equipment and its process data in form of a Resource Description Framework (RDF) triple store. For the Instances level, also a transformation method exists. Its goal is to enable an automated setup of the KBS's ABox by the information gained from the OPC UA and OBIX instances. The transformation processes at the Models level and at the Instances level are implemented in form of XSL transformations. The fact that there exist XML encodings for OBIX, OPC UA as well as for OWL suggests Extensive Stylesheet Transformation (XSLT) [34] as the method of choice. In this work, the Saxon XSLT and XQuery Processor[3] were used for XML schema and XSLT processing.

The two major tasks of such an XSLT are on one hand to amalgamate information from a BIM and automation technology in order to embed devices and datapoints into a building topology, and on the other hand to include address information in the resulting knowledge base individuals in order to make them transparent for physical access to live process data. In the following, this procedure is described in more detail. As shown in Figure

[3]https://sourceforge.net/projects/saxon/

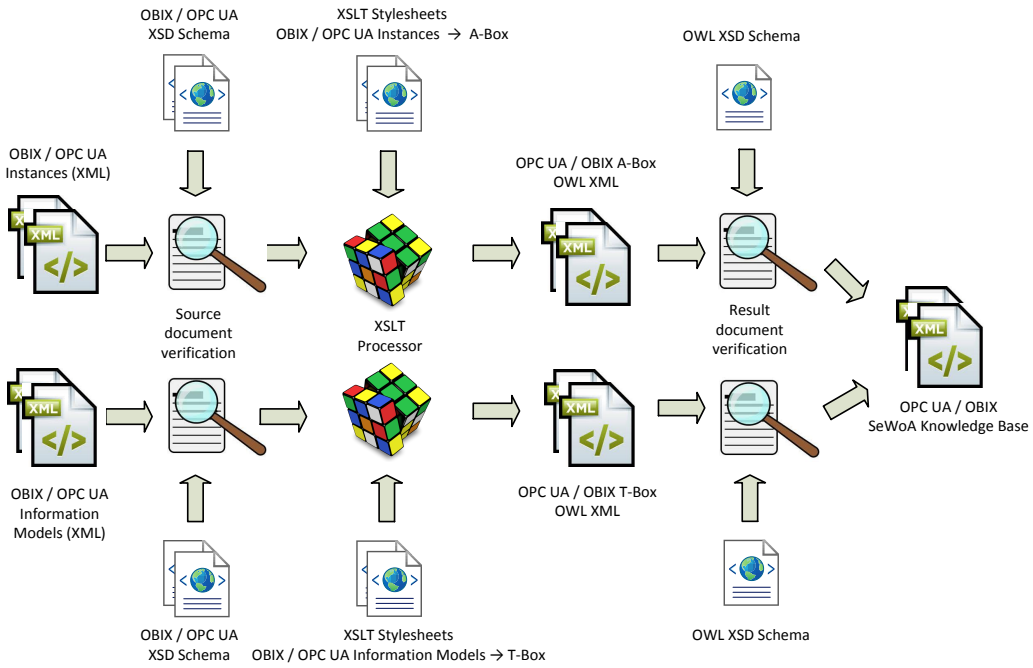

Figure 5.6: SeWoA information transformation

5.6, the transformation process is divided into two branches which are in correspondence to the two levels of the model hierarchy mentioned before. The lower branch outlines the XSLT from OBIX and OPC UA information models to the KBS's TBox. First, the source XML documents are validated against their XSDs (i.e., meta models) to ensure their syntactical correctness. Then, the actual transformation takes place. XSLT stylesheets have been defined for this purpose. The result documents are validated against the OWL XML schema. On Instances level, the transformation works analogously to the Models level. After validation against the OWL XML schema, the resulting OWL XML document containing the ABoxes are appended to the TBox OWL XML files generated before which results in OWL XML files containing the desired SeWoA knowledge base.

Figure 5.7 shows the overall concept and information flows including BIM, building automation systems configuration and runtime communication. Generally in this figure, information flow regarding configuration is illustrated by blue, curved arrows whereas the yellow, straight arrows represent runtime communication flow. The hierarchy of automation system components, also known as the automation pyramid, is illustrated in form of a large triangle. At the bottom of this hierarchy, there are on one hand Internet of Things (IoT) devices and on the other hand in form of a second, inner triangle, automation components of legacy building automation technologies like KNX, LonWorks and BACnet. A first level integration server on top of these legacy systems provides unified, technology-independent access by means of OBIX or OPC UA. Contrary to

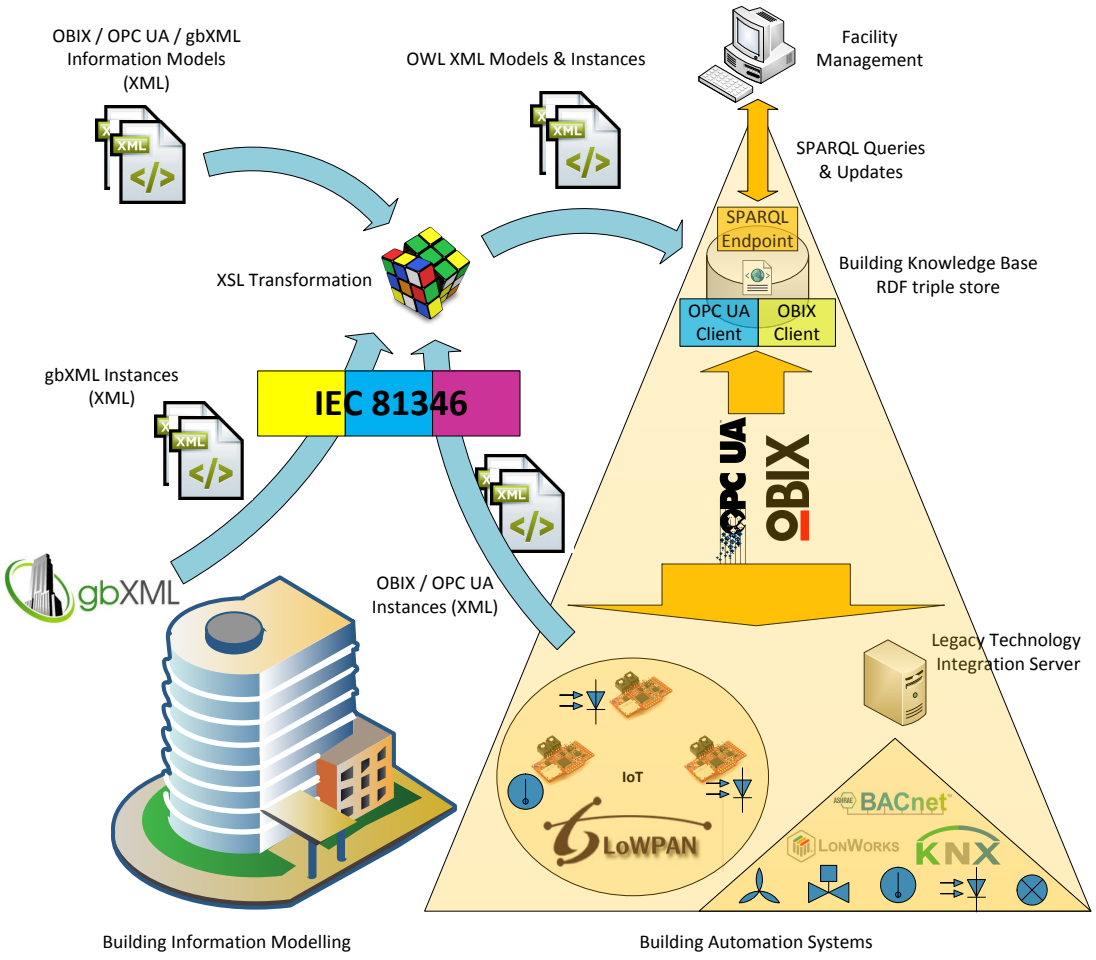

Figure 5.7: Information flow and prototype architecture

that, IoT automation devices natively provide Representational State Transfer (REST) interfaces like OBIX. The building knowledge base located on top of this hierarchy exchanges runtime information with the underlying systems via these OBIX or OPC UA interfaces, respectively. A SPARQL endpoint allows to apply semantic queries on the knowledge base.

The blue, curved arrows in the left part of Figure 5.7 show the information flow regarding static configuration. The central point is the XSLT following the transformation process described previously. Its output is transferred into the RDF triple store of the building knowledge base. The XSLT operates on Models and Instances level in order to provide not only an image of the actual building and its technical equipment to the knowledge base but also type information and type hierarchies which are essential for semantic reasoning on the knowledge base. Depending on which technology is in use, on model level OBIX, OPC UA and gbXML information models are taken as an input. The topology and other

information regarding the actual building under consideration are taken from a gbXML file. OBIX or OPC UA XML files reflecting the automation systems configuration, i.e., the automation systems instances, are taken from the respective integration servers (OBIX or OPC UA) or native IoT devices. This is illustrated in Figure 5.7 by the blue, curved arrow originating in the automation pyramid where OBIX an OPC UA devices are located. The IEC 81346 symbol overlaying the OBIX, OPC UA and the gbXML instances emphasises that both information sources must follow a common naming convention defined by this standard. Provided that this naming convention is taken into account, the XSLT is able to align the information gathered from BIM and automation systems engineering resulting in an embedding of the automation components in the building topology like shown in Figure 4.2.

5.3 Security infrastructure for integration technologies

Contrary to the classic OPC specifications, security is mandatory in OPC UA [32]. This should avoid bad experiences like made in the past with respect to developers of OPC products relying on the security mechanisms of the operating systems the OPC application runs on top of. This resulted in many systems being insecure and vulnerable. The security measures of OPC UA are unbundled of the operating system, so the developers of OPC UA products have almost full control over the security level of an application running in a distinct environment. For Web servers, the preferred use of HTTPS instead of HTTP communication is also usual practice today. The same should be the case when integrating Web service-based communication in automation systems.

OPC UA provides a very flexible security model that can be adapted to the desired use case. Also the Transport Layer Security Protocol (TLS), [61] which provides the security mechanisms of HTTPS, defines a variety of security profiles called cipher suites. There are different requirements on security, dependability and performance depending on the system in which these applications are embedded. In practice, the scale of the system, i.e., the number of OPC UA or Web service components installed as well as the human and financial resources available have an influence on the question, which approach leads to the optimal solution. The challenge is to find a trade-off between these requirements for each distinct environment. This circumstance is elaborated by comparing different ways of setting up such a secure infrastructure. Therefore, an introduction into the security architecture of OPC UA and TLS is given first. In OPC UA, communication between servers and clients is based on a session on top of a Secure Channel. TLS also establishes a session where an HTTP connection is set up on top. For both procedures, digital certificates are necessary. This leads to the question of how to establish trust relationships between these applications. Section 5.3.2 is dedicated to this topic. In the following Section 5.3.5, various software frameworks aiming at this target are presented. The chapter ends with a discussion (Section 5.3.6) where several methods of managing certificates based on different models of trust are briefly compared and evaluated.

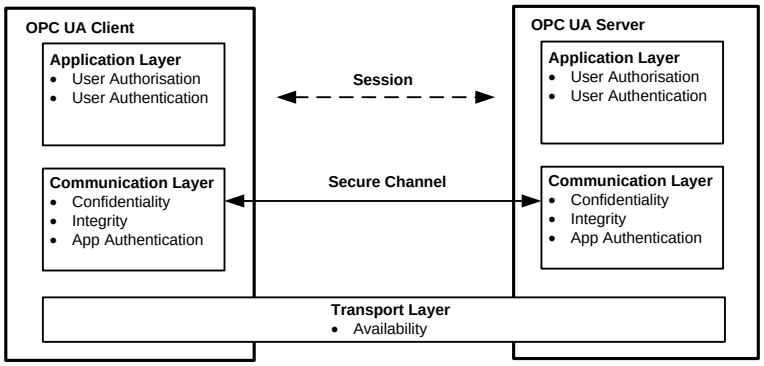

Figure 5.8: OPC UA security architecture ([32] Part 2)

5.3.1 Security architecture in OPC UA and Web services

Secure connections between an OPC UA client and a server are based on a three-layer architecture. This is shown in Figure 5.8. Connections established by the transport layer of the OSI Reference Model are based on server and client sockets. Here it is taken care of error detection and error recovery to achieve a reliable connection between the communication partners.

Based on this socket connection, a secure channel is opened by the communication layer of the OPC UA protocol stack. The communication layer is responsible for exchanging data in a secure way. Therefore, multiple security features are fulfilled: *Data integrity* is guaranteed by digitally signing the content transmitted. *Data confidentiality* is assured by encryption of data. In order to identify other applications and grant access, *authentication* and *authorisation* mechanisms are applied. For this purpose, ITU[4] X.509 certificates [166] are used. The same is an option for users instead of password-based authentication and authorisation.

The application layer on top of this architecture provides services for transmitting data, calling methods and exchanging configuration data between server and client in a session. Within a session, communication partners like users and certain products have to be *authenticated* and *authorised*. These tasks are managed by the OPC UA session services defined in the OPC UA specification Part 4 [32].

The TLS protocol has its roots in the commonly known Secure Socket Layer Protocol (SSL), which was released in 1994 by Netscape [167], [168]. In the end of the nineties, an adoption took place by the IETF TLS working group which resulted in the new protocol name “Transport Layer Security”. The current version is TLS 1.2 which is briefly described in the following. Figure 5.9 shows the TLS architecture embedded in the context of the OSI Reference Model. It is located at the Session Layer (Layer 5) on top of

[4]International Telecomunication Union, www.itu.int

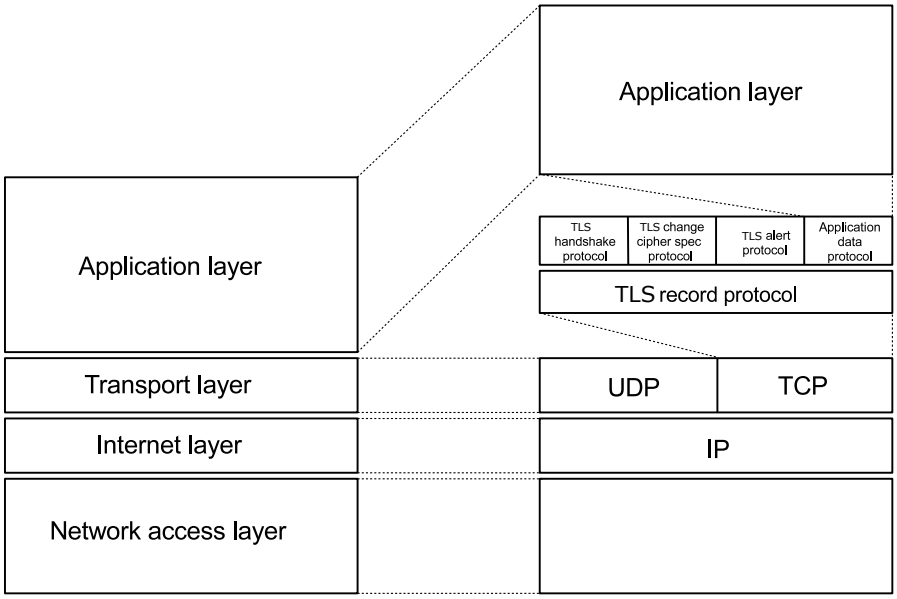

Figure 5.9: TLS architecture (adapted from [169])

the connection-orient communication of TCP/IP. It works as a transparent intermediate layer to application layer protocols like HTTP which allows its easy adaption.

TLS consists of two sub layers where the protocols from the upper layer (TLS Handshake Protocol, TLS Change Cipher Spec. Protocol, TLS Alert Protocol, TLS Application Data Protocol) are used for control purposes and the TLS record protocol at the lower layer. The latter handles the secure communication when the connection is established by assuring *Data confidentiality* and *Data integrity*. *Authentication* of communication partners is realised by the TLS Handshake Protocol using asymmetrical public key cryptography. For this purpose, like in OPC UA, in most cases X.509 certificates are used. However, also Pretty Good Privacy (PGP) keys [170] are supported. The TLS Application Data Protocol interlinks the data stream from the overlying application protocol with the TLS record protocol.

5.3.2 Certificate management

There exist different rules describing which entity creates certificates and how they are validated. These sets of rules constitute various trust models which are presented in the following. A digital certificate is an electronic document basically containing a *Public Key* (for deeper information about public key cryptography cf. [171]) and identity information about the owner of the certificate. A digital signature applied by a trusted third party, the *Certification Authority (CA)* or by the owner itself (*self-signed certificates*) is used to bind these two attributes together. This way, every other entity can check if the integrity of the certificate is preserved. Other attributes which are also included in a digital certificate are a serial number, a version field, the issuer of the certificate (the

CA or the owner) and the validity period. There may be also a field to be filled in with further information. The public key of a public/private key pair bound to the owner of the certificate is used for example for message encryption between a client and a server.

There must be found a way of organising the creation, distribution, validation and revocation of certificates. Some technical and organisational infrastructure is necessary to achieve this goal. Neither the OPC UA standard nor TLS define how such an infrastructure should look like. However, there exist some general concepts how to implement such an infrastructure.

The following section introduces these concepts where each of them is suitable for a different application scenario.

5.3.3 Trust models

Trust between End Entities (EEs) is achieved by either trusting in their associated certificates or trusting in a third party (*trusted third party*) that has previously authenticated the other entity. This is reflected in the different trust models introduced in the following.

A trust model can be organised in two ways, hierarchical by using one or more CAs or user-centric (decentralised) by applying the models of *Direct Trust* or *Web of Trust*.

- A *Web of Trust* gets along without any CA as a trusted third party. It only consists of EEs which make their own decision of whom to trust or not. This principle is applied in *PGP* and its open source siblings *OpenPGP* and *GnuPG*. This model does not scale well, since every EE needs to store a certificate of each EE it trusts. Also finding a trust path from one EE to another EE in big sets can consume a lot of computational power.

- The *Direct Trust Model* does not need any trusted third party, either. There are individual trust relations between the EEs. These trust relations have to be set individually for each EE. Therefore it does not scale well for big projects, either. This model is a very labour intensive one because usually the certificate distribution must be done manually (*out-of-band*). This method is only suitable for small environments but unreliable and inefficient for large scales.

- In a *Public Key Infrastructure (PKI)* there is one CA or even more CAs as trusted third parties in a hierarchy of inheritance which are organised like shown in Figure 5.10. A hierarchical organisation of CAs also results in a trust hierarchy, where a sub CA always trusts its super CA. Sub CAs can be assigned to particular units of an enterprise. This model scales well for big projects.

  Another way of organising CAs is a full-meshed architecture, which is a convenient approach if there is a lot of communication between different units of an enterprise. This way, trust paths are kept short since there is a direct relationship between the CAs instead of going up to the common root and back down the other branch. On

the other hand, path discovery may be more difficult since there may be multiple choices.

Each way of structuring a PKI has its advantages and disadvantages. A pro for a single CA is that it is easy to maintain. On the other hand, it has limiting effects on the size of the organisation. A multiple CA architecture scales well for big organisations but this advantage has to be bought by an administrative effort multiplied by the numbers of CAs present within the system.

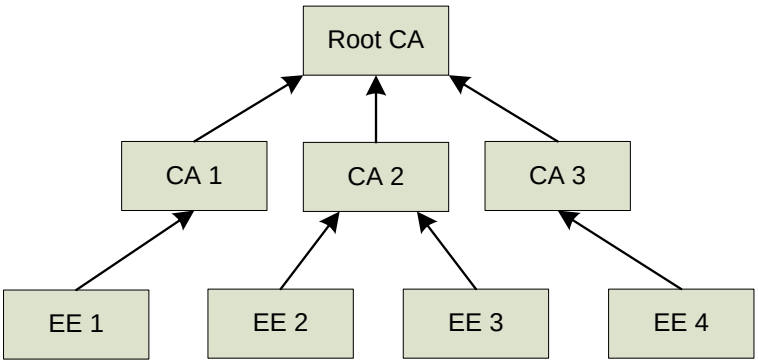

Figure 5.10: Hierarchical trust model [37]

5.3.4 Public key infrastructure

Concluding the different aspects of the particular trust models it can be claimed that a PKI is the most suitable one for the majority of automation systems integration applications using OPC UA or WSs. The direct trust model is only applicable for very small organisations and the Web of Trust does not scale well, either. Therefore, a closer look at the structure of a PKI is taken. A PKI consists of the following entities which are illustrated in Figure 5.11:

- An EE can be either an OPC UA product, an OPC UA user or a Web service component. It requests and uses the certificates issued by the CA.
- The CA is the trusted third party in a PKI. It generates documents based on the identity of end entities and the CA's private key. These documents are issued as certificates to other end entities.
- A *Registration Authority (RA)* is not an essential but optional component of a PKI. It performs tasks on behalf of the CA like verifying the identity of an EE and checking, if an EE is allowed to have a certificate, have its certificate renewed or revoked. After this verification, the RA forwards the EE's request to the CA.

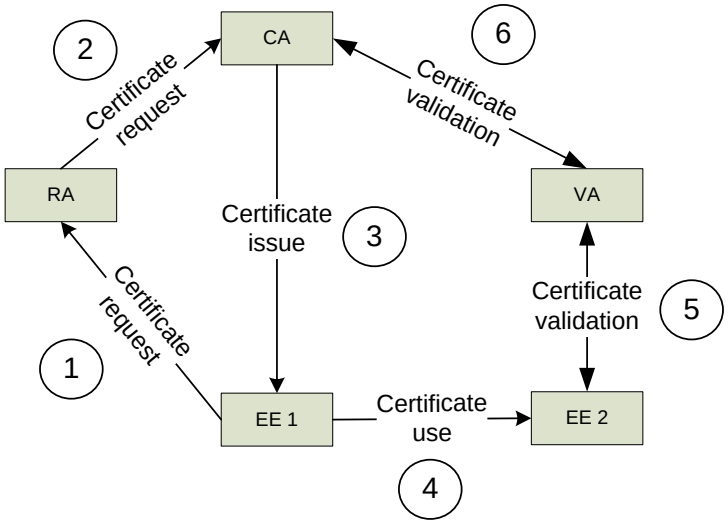

Figure 5.11: PKI enities and their interaction

- The *Validation Authority (VA)* has the purpose of validating certificates that EEs provide to it and returns the result of this calculation to the EEs. The validation process is performed by verifying the signature, checking the validity period and if the certificate has not been revoked. It must also be examined, if the usage of the certificate is within the specified purpose.

In a PKI, there exists a so-called *certificate lifecycle*. Figure 5.11 shows the different entities in a PKI and how they interact within a certificate lifecycle. It starts with the request of certificates by the EEs (1). After verification of the request by the RA, the request is propagated to the CA (2). The next step is to distribute the certificates among the EEs that issued the request (3). Now, the EEs (OPC UA applications, OPC UA users or Web service-enabled devices) can take the certificates for authorising and authenticating themselves or for message encryption (4). The communication partner requests a validity check by the VA (5) which in turn contacts the CA (6). Since there is a limited period of time in which certificates are valid, they need to be renewed or updated in case they are expired. If necessary, certificates can also be revoked. This is the case if e.g., the private key associated to the certificate is compromised or the certificate is not needed anymore. The usual approaches of setting up a PKI that manages the certificates lifecycle mainly differ in the methods of distribution and revocation.

There are the following ways of distributing certificates issued by a CA among the requesting EEs:

- Out-of-band: This method is performed manually by transporting the certificates on a storage medium (e.g., disk, USB-stick) to the EE or transferring it by email

to the target entity. Here it is imported into a local repository. This approach is an easy solution for small environments but does not scale well for big ones since it is labour intensive and unreliable because of the human component involved.

- Certificates can also be published in a central, well known, public repository like a *Lightweight Directory Access Protocol (LDAP)* [172] server. This can be seen as a Web server which provides access to a database containing certificates. The database content is controlled by a CA. This approach provides automatic download of certificates which makes it to a reliable solution. On the other hand, a single server is always in danger to be confronted with *Denial of Service (DoS)* attacks. Additional network traffic on an extra channel is also caused following this approach.

- In-band distribution means that an application-specific communication protocol is used for exchanging certificates. This way, no additional channel or protocol is necessary. *Secure/Multipurpose Internet Mail Extensions (S/MIME)*, TLS (cf. [173]) or even services from OPC UA can be used for this purpose.

For certificate revocation, there also exist multiple choices:

- Following the *offline* approach so-called *Certificate Revocation Lists (CRL)* are downloaded by the EEs from public, well-known locations like LDAP, FTP, HTTP servers at certain intervals. A CRL contains information about all the revoked certificates of a trust domain. It is signed by the CA that publishes the list. A CRL always reflects the past. It cannot provide any information about the current validity of a certificate. The currentness of a CRL depends on the update intervals, which are determined as a trade-off between network load and the desire of having an up-to-date CRL.

- The other choice is using *online* mechanisms to check if a certificate is valid. EEs connect to a service provider every time they use a certificate and thereby have their validity checked. A common way to achieve this is by using the *Online Certificate Status Protocol (OCSP)* [174]. It can provide real-time information, depending on the source of information the OCSP server relies on. Contrary to CRLs, also positive information can be disseminated about the validity of a certificate, i.e. an OCSP server can explicitly declare a certificate valid.

5.3.5 Applicable PKI frameworks for OPC UA and Web services

There is a variety of proprietary and open-source PKI frameworks available today, like OpenSSL[5], Microsoft Windows Server[6], OpenXPKI[7], VeriSign Managed PKI Services[8].

[5] http://www.openssl.org/
[6] http://www.windowsserver.com/
[7] http://www.openxpki.org/
[8] http://www.verisigninc.com/

Most popular OPC UA Software Development Kits (SDKs) published by the OPC foundation and the diverse software products based on these SDKs are written in .NET, Java and ANSI C. Therefore, a small summary of which framework can be used to operate OPC UA products in a trusted environment depending on the programming language and other criteria will be given.

The OPC Foundation released two tools that also can be used to manage OPC UA applications and certificates, the *UA Configuration Tool* and the *UA Certificate Generator*. Since there already exists a Whitepaper [175] giving a description of these tools and a guideline about the administrative procedures that lead to a secure environment for OPC UA applications, there is no further discussion about these tools in this work.

Windows Server, .NET

A Windows based PKI is a convenient commercial solution available for a reasonable price. Many enterprises use a Windows environment anyway, so it is only about using additional features of Windows Server and the client operating systems. The Windows *ActiveDirectory Certificate Services* exist since Windows NT 4.0 and can therefore be considered as very mature.

Windows Server machines act as the CAs in a PKI. RAs can be set up by also Windows based *Internet Information Services (IIS)* Web servers. This allows *Web Enrolment*, i.e. an automatic way of certificate dissemination among the EEs. User identity information is also gathered by an IIS server and verified by using ActiveDirectory. LDAP services are also available via ActiveDirectory.

A step by step guideline of how to setup a Windows based PKI is provided by Microsoft TechNet [176].

VAs are represented by the local Windows Certificate Stores of each machine, which provide an abstract, unified way to access certificates. OPC UA applications based on .NET must implement routines to access certificates from the certificate store. An article which describes how to deal with certificates in .NET applications and also containing example code can be found at the Microsoft documentation website [177].

OpenSSL

OpenSSL is an open source toolkit written in C. It implements the *Secure Sockets Layer* and the *Transport Layer Security* protocols as well as a general purpose cryptography library. It is released under an Apache-like licence. Supported platforms are UNIX-like operating systems and Windows.

Besides the C libraries supporting cryptographic services, OpenSSL also provides a command-line program called *openssl* with all the functionalities necessary for managing a PKI. For this purpose, the toolkit must be installed on every machine of the environment. Issuing certificate requests, the generation of RSA keys and X.509 certificates as well as commands to revoke certificates and to generate a CRL are the main features of this tool.

All these steps can be performed in a manual way where the administrator acts as the CA and the RA. This approach is suitable for small environments. In order to handle certificate management in bigger projects, script based execution of these operations is recommendable. Further information containing a step-by-step instruction of how to set up a PKI with OpenSSL can be found at the Symantec webpage [178].

OpenXPKI

OpenXPKI is an open source software implementation targeting from small installations to enterprise-level (large-scale) PKIs. It is designed for Unix-like operating systems and is released under an Apache licence. The key features of OpenXPKI are the support of multiple CA instances on a single application instance in order to set up a trust hierarchy and the capability of full automatic CA rollover. This way, continuous operation of the PKI is assured without administrator intervention, in case one CA certificate expires and another CA has to take over. High flexibility is achieved by an XML-file controlled workflow engine that allows extending the basic PKI operations.

An OpenXPKI installation can act as a CA, a RA or an EE, depending of its configuration. Certificates, private keys and revocation information are stored in a database system that can be chosen out of the most popular ones like MySQL and Oracle. Documentation for developers is provided on the project Webpage [179].

This solution will mainly be applicable for large scale installations because it causes quite an effort to set up and configure an OpenXPKI based PKI. This work is best done by professional developers. The complexity of this framework is a result of its high flexibility and modular design.

Java

There is a powerful security API delivered with the Java SDK. It also includes classes (`java.security`) related to PKI applications. Their functionalities encompass support for X.509 certificates, CRLs and PKIX-compliant [180] certification path building and validation [181]. Classes that provide a key store (a secure repository for cryptographic keys) and a certificate store are also available. This makes the Java security API mainly interesting for the use in EEs of a PKI.

Additionally, there is a command-line program named `keytool` which can be used for creating and managing key stores. The main features of this tool are:

- Create public/private key pairs and self-signed certificates
- Display, import and export X.509 certificates stored as files
- Issue PKCS#10 [182] (a standardised message format) certificate requests to be sent to CAs
- Import certificate replies obtained from the CAs as responses to certificate requests

- Designate public key certificates as trusted

More details about this tool and the Java security API can be found in the Java SE documentation available at the Oracle Website [181].

5.3.6 Discussion

The question about the best-suited model for a given application is a hard one to find an explicit answer on. Moreover, the development of proper security metrics which would provide methods for quantitative comparison of different approaches is still an ongoing field of research [183]. The likelihood is high that it is not possible to find metrics for every security aspect. The decision on the particular trust model implementation does not only depend on hard facts like size, structure and the equipment already in use but is also based on personal taste. One may prefer the use of free software where it is often more effort to set up a working system. Also hiring external experts for this purpose may be necessary. Another one favours commercial software which is usually bought as an out-of-the-box product including support. This again results in less effort for the internal personnel.

However, there are several criteria that may lead the decision which solution to apply to a distinct direction:

The *budget* available has a limiting influence on this choice. A single CA PKI is naturally cheaper since there is less maintenance effort compared to the multiple amount of effort caused by a hierarchical or a full-meshed architecture. Higher maintenance effort results in higher personnel costs. For small applications also the direct trust model can be feasible. This represents the cheapest solution since there is no extra security-related equipment necessary.

The availability of *human resources* has a similar effect on the decision upon a trust model. For small organisations, a single CA architecture or also the direct trust model will be sufficient. This can be seen from two perspectives: little human resources operating a small set of machines in a trusted environment issue little requests for certificates. This keeps on the other side the infrastructure small, i.e., the equipment to manage these requests and the personnel maintaining the equipment. The opposite applies to large scales, which will result in an architecture with multiple CAs. The decision whether to deploy a hierarchy of CAs or a mesh of CAs depends on other criteria.

Dependability may be a goal to achieve in case danger threatens humans or material if some equipment is not available caused by a faulty trust infrastructure. In a single CA or hierarchical CA architecture, there is always a single point of failure: The single CA or the root CA, respectively. On the contrary in a meshed architecture no fail of the whole PKI takes place if one CA is compromised or out of service. Only the users or EEs are affected which have a trust relationship to the respective CA. Recovery is also easier since a new certificate only needs to be distributed among these few affected EEs.

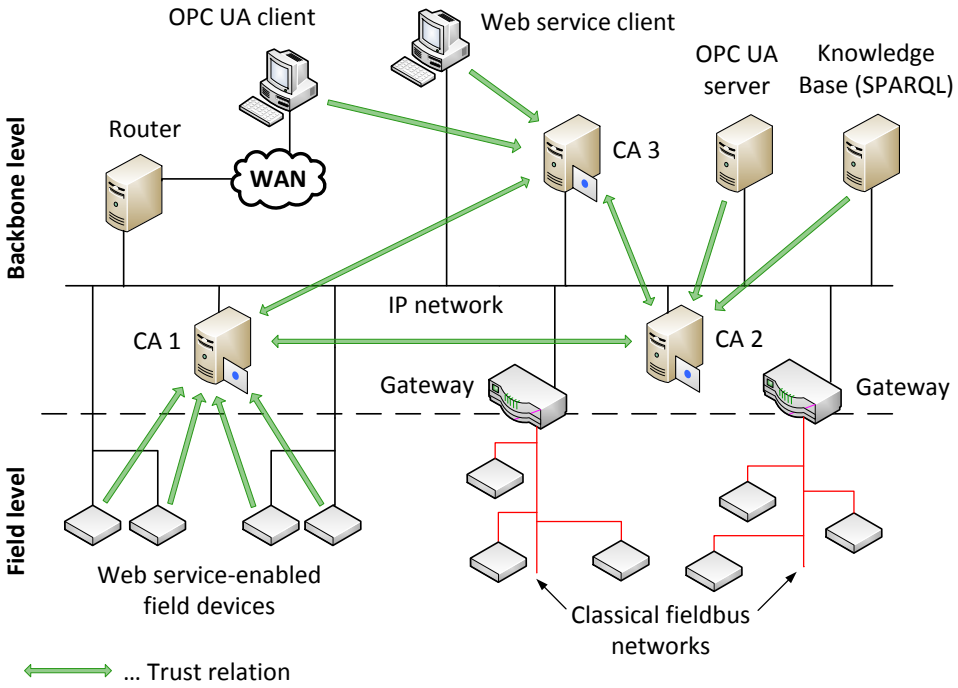

Figure 5.12: Two-tier automation network establishing a hybrid trust model

It is finally convenient to map the *structure* of the organisation to the model of trust. This will mostly be applicable if it is already clear that a single CA architecture or the direct trust model will not fulfil the requirements because a distinct size of the organisation is already exceeded. If a hierarchical structure is inherent with the organisation, i.e., there is mainly communication between sub units and super units, then the trust relationship should be organised this way. On the other hand, if there is also considerable communication between units on the same level, the meshed PKI architecture is probably the right choice.

An idea, how trust relations can be organised in a real-world automation system is given in Figure 5.12. It illustrates a two-tiered automation hierarchy consisting of a field level and a backbone level, where data from the lower tier is aggregated and a common interface to management and enterprise applications is provided.

In this example, OPC UA and Web service-enabled devices are deployed at both tiers of the automation system. An OPC UA server is located at the backbone level to enable standardised and uniform access to process data of the lower level, like shown in the right part of Figure 5.12. A KBS also located at the backbone level makes a holistic view on the integrated systems available via a SPARQL endpoint. To reflect the current trend leading to use WSs all down to the field level, the setup also contains a network of Web service-enabled field devices in the left lower part of the figure.

The trust model used in this example follows a hybrid approach. The OPC UA server, the Web service-enabled field devices as well as the knowledge base (the EEs) are part of PKIs and receive their certificates from different CAs. There is one CA for each group of devices: one for the Web service-enabled field devices, one for the OPC UA server and the SPARQL endpoint of the knowledge base, and one for the management workstations, i.e., the OPC UA and Web service clients. This way, a separation between these groups regarding certificate management is achieved. This provides on one hand a distinct level of dependability and keeps on the other hand the number of EEs per CA limited. In order not to lose dependability gained by the multiple CA approach, there is no single root CA in this system but trust between the CAs is established by implementing a full-meshed architecture. Yet, a compromise is made with respect to scalability of the number of CAs since the number of trust relationships between CAs grows polynomially.

5.4 Results

In the first section of this chapter, a concept for semi-automated information processing in building automation engineering has been proposed. It defines an approach of a consistent workflow from building and automation system planning information to the engineering tool of the popular building automation standard KNX. Furthermore, this concept is pursued for first level and second level integration platforms. A way to automatically instantiate an OPC UA-based middleware on this planning data and additional device and datapoint semantics from the KNX application model was shown. In a final step, a transformation of this information to a KBS realising second level integration was elaborated.

The second section of this chapter presents different methods of certificate management (trust models) in OPC UA and Web service applications. As a reason of the limited scalability of the Web of Trust and the Direct Trust Model, for medium and large scale environments the PKI evaluated as the most convenient approach of managing certificates. Nevertheless, the Direct Trust Model can be an option for low-scale automation system in which a very limited number of OPC UA or Web service devices are installed. The choice between the presented PKI frameworks on one hand depends on the operating system preferred or already in use in the environment, respectively. There are frameworks available for Windows, Unix-like platforms as well as the platform-independent Java security API. On the other hand, different features provided by the distinct frameworks have an influence on this decision. Finally, the structure of the control network where OPC UA and Web service applications are installed, the resources available and the requirements on dependability affect the strategy of managing certificates.

CHAPTER 6

Conclusion and outlook

6.1 Results

The starting point of this thesis was the definition of problems arising when it comes to integration in the highly heterogeneous landscape of modern Building Automation System (BAS). The central question to be answered was:

Which state-of-the art technologies are applicable and provide the desired, tailored expressiveness for each level of integration? Moreover, how can a secure operation and an efficient engineering workflow be assured?

6.1.1 Contributions in relation to the research question

A first step towards answering this question was done in Chapter 1 where an analysis of existing knowledge representation languages has been carried out. This analysis resulted in a number of integration technology candidates which in principle fulfill the requirements on their expressive power with regard to the defined first level and second level integration intentions. One of these candidates, i.e., OPC Unified Architecture (OPC UA) for first level integration and the Semantic Web technologies Web Ontology Language (OWL) and SPARQL Protocol and RDF Query Language (SPARQL) for second level integration are instrumentalised in the following Chapters 3 and 4 to create concrete information representations suitable for these two levels. Finally, engineering and security aspects were addressed in Chapter 5. Hereby, the following defined requirements were taken into account:

- ***Requirement 1**: An infrastructure shall be created which integrates BAS technologies in a way that seamless information exchange is possible independent of the automation technology.*

OPC UA information models have been introduced in order to integrate the building automation technologies Meter Bus (M-Bus) (Section 3.2), KNX (Section 3.3) and Building Automation and Control Network (BACnet) (Section 3.4) into a homogenised layer of information representation. This layer constitutes the proposed first level integration. At the time of publication, the mappings presented had and still have a high degree of novelty. In literature, no comparable approaches with the focus on building automation integration exist.

- ***Requirement 2**: The infrastructure from Requirement 1 shall be extended such that access to systems from both industrial and building automation domains is enabled in a unified way.*

  OPC UA information models dedicated to use cases bridging the domains building automation and factory automation have been defined in Section 3.5. This includes an architecture of integration servers allowing to seamlessly access information from automation systems of both domains. The idea of cross-domain integration by means of OPC UA has not been taken up by the community so far and therefore no methods exist comparable to the ones presented in this work.

- ***Requirement 3**: The integration infrastructure shall comprise spatial and topological information associated with runtime and device information.*

  The defined OPC UA information models for KNX (Section 3.3) and especially the cross-domain models in Section 3.5 explicitly include perspectives representing topological information on buildings and plants. An association between these topological models and datapoints of the underlying processes is established. The modelling approaches in currently existing in literature mainly focus on a device-centric view. One work which also regards topological aspects is [184] where a meta model towards a Web service representation of KNX systems is introduced. However, this publication has been published several years after the author's ones regarding this topic.

- ***Requirement 4**: A way of representation shall be established with the capabilities to map a holistic view of a site including building information and the embedded automation systems. In order to cope with the expected large scale of information, this representation shall also be able to process semantic queries.*

  In order to realise this second level integration where automation systems integration together with building information models are lifted into a common context, OWL ontologies have been defined, partly by the extension of existing ones (Section 4.2). The novelty of the resulting holistic building knowledge base is the seamless integration of runtime data from the underlying first level integration layer. A SPARQL endpoint allows to issue comprehensive queries to the knowledge base. In literature, a variety of building automation ontologies exist. However, the unique feature of the second level approach is the integration of runtime data by

establishing a logical link between the state value instances in the knowledge base and actual datapoints in the underlying automation systems.

- ***Requirement 5**: A concept shall be defined which supports building automation and integration engineering by avoiding manual information transitions between the involved intermediate steps.*

  Several approaches facilitating the engineering of automation systems as well as automatising the instantiation of integration services are proposed in this thesis. In Section 3.3.3, a method is sketched which allows to automatically instantiate an OPC UA server's address space using the information provided by a KNX engineering project. A refinement of this attempt is presented in Section 5.2 which is dedicated to semi-automated engineering procedures at multiple levels of the automation hierarchy. This encompasses field device engineering by means of a case study on KNX (cf. Section 5.2.1) and automated first level integration (cf. Section 5.2.2), i.e., OPC UA-based middleware engineering. Finally, an approach of automatically instantiating the knowledge base described in Section 4.2 using information available in the first level integration layer is described. This transformation process is described in detail in Section 5.2.3. As the analysis of related scientific work showed, a considerable number of publications on engineering support exists. Their main focus is on field device engineering where far-advanced approaches have been developed. The novelty of the engineering workflow introduced in this thesis is that it encompasses field devices and the two proposed integration layers. Information defined once is reused throughout the whole process.

- ***Requirement 6**: For the integration infrastructure to be designed, state-of-the-art security measures shall be deployed. This especially includes measures for establishing trust relationships between the involved components.*

  All integration standards used in this work allow to realise standard security features like data confidentiality and integrity as well as device and user authentication by using public key cryptography. These mechanisms base on X.509 certificates which additionally provide the preliminaries to establish trust by means of a Public Key Infrastructure (PKI). As alternatives to a PKI, other state-of-the-art trust models are addressed in a discussion about their applicability in BASs integration in Section 5.3. The differentiation to related work in this field is the strong focus of this section on the context of an integration infrastructure like proposed in the previous chapters of this work. Special circumstances like the hierarchical system architecture and the different roles of integration components resulting thereof are regarded.

6.1.2 Contribution to standardisation activities: OPC UA in the KNX Web service specification

In order to close the gap between KNX installations and the IT world by using Web services (WSs), the KNX Association[1] started an initiative which aimed at developing the so-called *KNX WS specification*. In the course of these activities, a standardised Web service interface was defined for KNX installations. It allows communication between KNX and management IT systems by a gateway implementing this interface. This section focuses on the OPC UA part of this KNX WSs specification which was mainly developed by the author. However, the big picture of this specification is presented in [184].

The general approach of this KNX WS gateway concept is to expose the necessary pieces of information inherent in a KNX network to local or remote client devices using Web service technologies. This information includes

- Building topology,
- Device model (properties, meta information),
- Live datapoints, i.e., the process image of the underlying KNX installation.

It is further enriched with semantics including a type system and physical units. It is intended to gain this information from the KNX Engineering Tool Software (ETS) in a mostly automated way. To this aim, ETS projects which in most cases exclusively contain the complete engineering information of a KNX installation are extracted. An exception thereof are KNX devices which rely on vendor-specific ETS plugins for configuration. The concepts of the KNX WS specification are very closely related to the way how information about a KNX installation is represented in the ETS.

The KNX WS specification defines an abstract tag vocabulary-based meta model with the ability to express the information mentioned above. It is independent of an actual technology implemented by an actual KNX WS gateway. In a second step, transformation rules from this meta model to the information models of the most common integration technologies OPC UA, Open Building Information Xchange (OBIX) and BACnet/Web Services (BACnet/WS) models are defined. By this way, the KNX WS specification stays highly maintainable and extensible since changes only have to be applied to the tag vocabulary which are automatically propagated to the derived Web service technologies by the defined transformation rules.

For the OPC UA incarnation of the KNX WS meta model, the topological aspect of the KNX information model is reflected by an arbitrary deep hierarchy of the built-in ObjectType `FolderType`. By means of such a structure of folders, it is possible to reflect a building layout consisting of floors, rooms and stairways. For the device and datapoint aspects of a KNX installation, it is drawn on already published high-level OPC

[1] https://www.knx.org

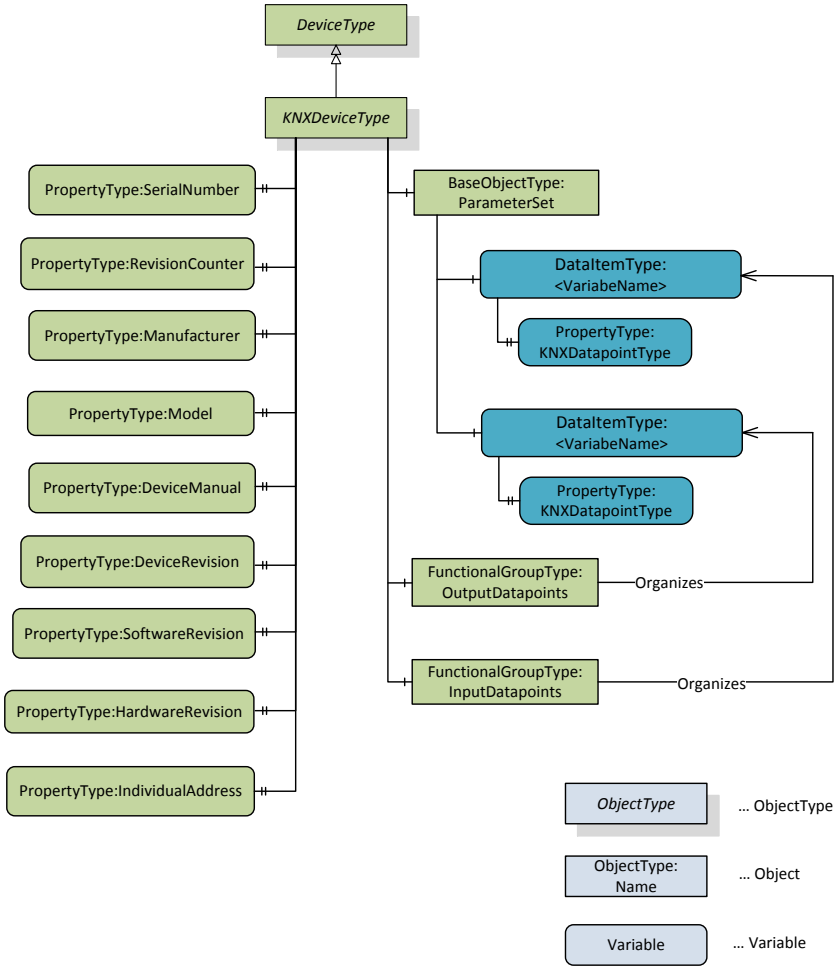

Figure 6.1: OPC UA KNX device model

UA information models. The already mentioned OPC UA for Devices (DI) specification provides a very detailed concept for modelling field devices and is therefore well-suited for the KNX device model. Figure 6.1 shows the abstract type definition of a `KNXDeviceType` which is a subtype of the abstract `DeviceType` defined in the DI specification. It therefore inherits its structure from its supertype which is extended by KNX-specific properties. The properties on the left branch of the `KNXDeviceType` definition provide static information about the device like a serial number, hardware or software revision or manufacturer information. The `IndividualAddress` Property is an extension to the `DeviceType` and reflects the physical, i.e., individual address of a KNX device. The `ParameterSet` Object on the right side exposes the actual datapoints of the device. Their number and types naturally vary for each concrete KNX device for which this abstract definition provides the necessary freedom. A `KNXDatapointType` property informs about the underlying KNX Data Point Types (DPTs). The `OutputDatapoints`

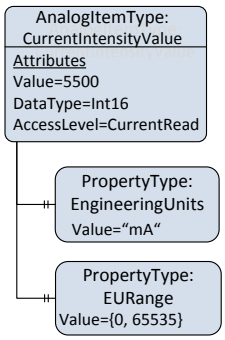

Figure 6.2: Datapoint example - AnalogItem

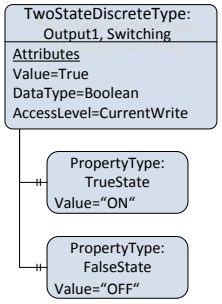

Figure 6.3: Datapoint example - TwostateDiscreteItem

and `InputDatapoints` Objects assign an I/O direction to the datapoints.

In order to model the semantics of KNX datapoints defined in their according type definition (KNX DPTs), the OPC UA Data Access model [146] is used. It defines a number of general purpose VariableTypes for modelling analogue and discrete datapoints. Besides the actual value, additional semantics like datatype, access level, unit and value range are provided. The Figures 6.2 and 6.3 show instance examples of a `CurrentIntensityValue` and a switching value `Output1, Switching` datapoint. When instantiating a concrete KNX device, DataItems like shown in these examples are inserted beyond the `ParameterSet` Object of the `KNXDeviceType`.

Figure 6.4 shows the runtime interaction between the gateway and an OPC UA client. When a client wants to read the current value of Node attribute (e.g., the current value of a datapoint), it issues a *Read* service call to the KNX OPC UA gateway. The gateway internally determines if the attribute to read is a KNX datapoint or statically defined value in the OPC UA server's address space (e.g., a DisplayName or a Description). If the read access regards a KNX datapoint, the gateway resolves the request to an internally stored KNX group address and puts a *Group Value Read* request to the KNX

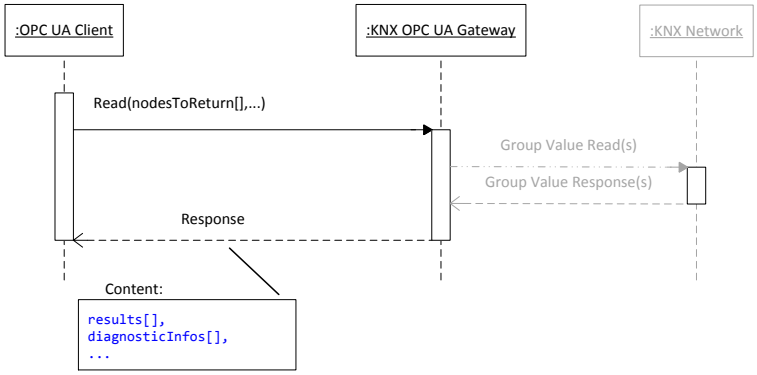

Figure 6.4: Read transaction

network. After the KNX network has responded, the gateway can return the OPC UA read response containing the DataValue and additional diagnostic information to the client. This diagnostic information contains a time stamp, status information about the gateway or error codes in case the request did not success.

6.2 Open issues

6.2.1 First level integration

Regarding the mapping of the M-Bus application model to OPC UA, there are still open issues that need to be elaborated in the course of future work. One is the service mapping between M-Bus and OPC UA. Also the inclusion of historical measurement values provided by M-Bus into the OPC UA information model needs to be discussed. The OPC UA Historical Access specification is considered to have appropriate mechanisms to address this problem. Another feature of M-Bus enabling master devices to apply configuration changes on meters has been unattended, yet. Regarding this manner, it needs to be discussed if this is a reasonable use case for management applications or if collecting metering data is sufficient. In order to evaluate the feasibility of the concepts introduced by this modelling approach, a prototype needs to be implemented. One way for physical interaction with the bus is the use of stock M-Bus to EIA RS-232 converters.

Another issue regards the homogenisation of the presented OPC UA information models for KNX. The resulting representation should combine the introduced topological models on one hand and also base its device model completely on the DI specification. Also standardisation activities towards an OPC UA companion specification for KNX could be considered. This should be done in alignment with the OPC UA companion specification for BACnet currently under development.

6.2.2 Second level integration

Future work on the presented second level integration by a Knowledge-Based System (KBS) could additionally consider the temporal aspect of runtime data by means of adding time stamps to the datapoint model. Furthermore, it is imaginable, as shown in [185], to represent time series of data within the knowledge base which would be valuable for use cases regarding diagnostics. Another challenge is to wrap the complexity of SPARQL queries and hereby increase usability for non-experts. Natural language-to-SPARQL transformations like introduced in [186] could be a suitable method to this. To this aim, also open source implementations exist[2].

6.2.3 Building automation systems and security engineering

The presented semi-automated KNX engineering attempt also has one drawback, namely the remaining manual interaction part during ETS project engineering. This issue might be overcome by integrating the ideas from [126] at this point of the workflow. Also the implementation of prototypes performing the proposed automated engineering processes is still pending. Utilising the present engineering information, another future capability is the automated generation of visualisations at Supervisory Control and Data Acquisition (SCADA) level. To this aim, the current workflow needs to be extended towards user interface markup languages like defined by the World Wide Web Consortium (W3C) or available for Java applications.

Since security engineering in automation systems integration is a very practice-related topic, experiences made by companies operating in this field could be taken into account to approve the results of the discussion carried out. A quantitative survey evaluating the feedback given by automation systems integrators and end-users could give more insight into this topic.

[2] https://github.com/nmvijay/freya

List of Figures

Acronyms

ABox Assertional Box. 20, 35, 36, 103, 104, 123, 125, 126

AEC Architecture, Engineering and Construction. 38

API Application Program Interface. 24, 53, 81, 110, 112, 120, 136, 137, 139

ARCNET Attached Resource Computer Network. 3, 53

ASHRAE American Society of Heating, Refrigerating, and Air Conditioning Engineers. 52

BACnet Building Automation and Control Network. 5, 52–55, 74–83, 98, 99, 142, 147, 150

BACnet/WS BACnet/Web Services. 24, 48, 144

BAS Building Automation System. xi, 1, 3–5, 9–11, 14, 15, 37, 47, 49, 84, 86, 92, 115, 116, 141, 143, 149

BEMS Building Energy Management System. 4, 10

BIM Building Information Modelling. xi, 38, 41, 101, 106, 112, 117, 120, 123, 125, 126, 128

BMS Building Management System. 4, 5, 38, 44, 45, 63, 66, 74, 102, 110, 112, 119

CA Certification Authority. 130–139

CAE Computer Aided Engineering. 43

CAEX Computer Aided Engineering Exchange. 38, 43

CO Communication Object. 51, 62–64, 67, 71, 72, 88, 92

CoAP Constrained Application Protocol. 19, 31, 40, 106

COM Component Object Model. 24

CORBA Common Object Request Broker Architecture. 7

CRL Certificate Revocation Lists. 134–136

DCOM Distributed Component Object Model. 7, 24

DDC Direct Digital Control. 2, 3, 5

DI OPC UA for Devices. 56, 85, 86, 88, 123, 145, 147

DIB Data Information Block. 50, 56

DoS Denial of Service. 134

DPLL Davis-Putnam-Logemann-Loveland. 22

DPT Data Point Type. 51, 60, 145, 146

DRH Data Record Header. 50

DSL Domain Specific Language. 44

DTLS Datagram Transport Layer Security. 31

EDDL Electronic Device Description Language. 40

EE End Entity. 131–137, 139

EMS Energy Management System. 3

ERP Enterprise Resource Planning. 6, 40

ETS Engineering Tool Software. 51, 60–71, 73, 74, 99, 117, 119–123, 144, 148–150

EXI Efficient XML Interchange. 40

FB Functional Block. 44, 51, 60, 62, 63

FDI Field Device Integration. 40

FDT Field Device Tool. 40

gbXML Green Building XML. 38, 39, 102, 104, 106, 117, 121, 123, 127, 128

GIOP General Inter-ORB Protocol. 19

GO Group Object. 51

GUI Graphical User Interface. 4, 40, 82, 98

HBA Home and Building Automation. 51

HLK Heizungs-, Lüftungs- und Klimatechnik. ix

HMI Human Machine Interface. 3

HVAC Heating, Ventilation and Air Conditioning. xi, 2, 3, 10, 37, 43, 61, 62, 83, 84, 90, 104

IFC Industry Foundation Classes. 38, 41, 102

IIS Internet Information Services. 135

IoT Internet of Things. 40, 44, 102, 104, 109, 110, 126–128

JSON JavaScript Object Notation. 28, 33, 37

KBS Knowledge-Based System. 21, 35, 90, 102–104, 106, 112, 123, 125, 126, 138, 139, 148

LDAP Lightweight Directory Access Protocol. 134, 135

M-Bus Meter Bus. 5, 10, 49, 50, 55–59, 96, 98, 142, 147, 149

M2M Machine-to-Machine. 11, 14, 28, 33

MDH Manufacturer Data Header. 50

MEP Mechanical, Electrical and Plumbing. 38

MES Manufacturing Execution System. 6, 40

MS/TP Master-Slave/Token Passing. 53

NIST National Institute of Standards and Technology. 9

OASIS Organization for the Advancement of Structured Information Standards. 28, 123

OBIX Open Building Information Xchange. 18–21, 24, 28–31, 40, 48, 102–104, 106–110, 112, 113, 116, 123, 125–128, 144, 149, 150

OCSP Online Certificate Status Protocol. 134

OMG Object Management Group. 18

OPC OLE for Process Control. 24

OPC UA OPC Unified Architecture. ix–xi, 18–21, 24–28, 40, 43, 48, 49, 55–57, 60–75, 77–79, 81–88, 90–92, 95, 96, 98, 99, 102–104, 106–110, 112, 116, 117, 120, 122–135, 138, 139, 141–144, 146, 147, 149, 150

OWL Web Ontology Language. ix, xi, 21, 32, 34, 35, 40, 42–44, 105–107, 109, 110, 123, 125, 126, 141, 142

PGP Pretty Good Privacy. 130, 131

PKI Public Key Infrastructure. 131–139, 143, 150

PLC Programmable Logic Controller. 2, 6, 40, 43

PLCopen OPC UA Information Model for IEC 61131-3. 86, 88

PTP Point-To-Point. 53

RA Registration Authority. 132, 133, 136

RDF Resource Description Framework. 32–35, 37, 41, 104, 105, 108–110, 112, 125, 127, 149

RDFS RDF Schema. 32–34

REST Representational State Transfer. 31, 127

S/MIME Secure/Multipurpose Internet Mail Extensions. 134

S4AC Social Semantic SPARQL Security for Access Control. 104

SAT Satisfiability. 20, 22

SCADA Supervisory Control and Data Acquisition. 4, 6, 119, 148

SDK Software Development Kit. 20, 81, 96, 98, 135, 136

SOA Service Oriented Architecture. 48

SOAP Simple Object Access Protocol. 31

SPARQL SPARQL Protocol and RDF Query Language. 21, 22, 36, 37, 40–42, 44, 102–104, 106, 109–112, 116, 127, 138, 139, 141, 142, 148

SSL Secure Socket Layer Protocol. 129

TBox Terminological Box. 20–22, 35, 102, 104, 123, 125, 126

TLS Transport Layer Security Protocol. 31, 128–131, 134

UML Unified Modeling Language. 18, 19, 43

VA Validation Authority. 133

VIB Value Information Block. 50, 56, 58

W3C World Wide Web Consortium. 31, 32, 35, 36, 123, 148

WS Web service. x, xi, 6, 7, 19, 24, 47, 48, 106, 116, 122, 123, 132, 138, 144

XACML eXtensible Access Control Markup Language. 31

XSD XML Schema Definition. 19, 33, 39, 82, 108, 120, 123, 126

XSLT Extensive Stylesheet Transformation. 19, 103, 109, 112, 125–128

Bibliography

[1] "Johnson Controls Automatic Temperature Control System - A Historic Mechanical Engineering Landmark," American Society of Mechanical Engineers (ASME), May 2008.

[2] R. Panke, *Energy Management Systems and Direct Digital Control.* Fairmont Press, 2002.

[3] "Building Automation and Control Systems (BACS) – Part 2: Hardware," ISO/DIS 16484-2-2016, 2016.

[4] W. Kastner and G. Neugschwandtner, "Datenkommunikation in der verteilten Gebäudeautomation," *Bulletin SEV/VSE*, August 2006.

[5] G. Levermore, *Building Energy Management Systems: An Application to Heating, Natural Ventilation, Lighting and Occupant Satisfaction.* Taylor & Francis, 2013.

[6] T. Sauter, S. Soucek, W. Kastner, and D. Dietrich, "The evolution of factory and building automation," *IEEE Industrial Electronics Magazine*, vol. 5, no. 3, pp. 35–48, 2011.

[7] "Communication systems for meters - Part 2: Wired M-Bus communication," DIN EN 13757-2:2016, 2016.

[8] "Communication systems for meters - Wireless mesh networking for meter data exchange," EN 16836, 2016.

[9] "KNX Specification 2.1," October 2013.

[10] "Open Data Communication in Building Automation, Controls and Building Management - Control Network Protocol - Part 1-5," ISO 14908-1 - ISO 14908-4, 2014.

[11] "Building Automation and Control Systems (BACS) – Part 5: Data Communication Protocol," ISO 16484-5, 2016.

[12] W. Kastner, G. Neugschwandtner, S. Soucek, and H. M. Newman, "Communication Systems for Building Automation and Control," *Proceedings of the IEEE*, vol. 93, no. 6, pp. 1178–1203, Jun. 2005.

[13] A. Kalogeras, J. Gialelis, C. Alexakos, M. Georgoudakis, and S. Koubias, "Vertical integration of enterprise industrial systems utilizing web services," *IEEE Transactions on Industrial Informatics*, vol. 2, no. 2, pp. 120–128, May 2006.

[14] T. Sauter, "Integration aspects in automation - a technology survey," in *10th IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 2. IEEE, 2005, pp. 9–17.

[15] "OPC Foundation," last visited March 2018. [Online]. Available: www.opcfoundation.org

[16] W. Granzer and W. Kastner, "Information modeling in heterogeneous Building Automation Systems," in *9th IEEE International Workshop on Factory Communication Systems (WFCS)*, May 2012, pp. 291–300.

[17] "Common Object Request Broker Architecture Specification Version 3.3," Object Management Group, 2012.

[18] S. Soucek and D. Loy, "Vertical Integration in Building Automation Systems," in *5th IEEE International Conference on Industrial Informatics (INDIN)*, vol. 1, June 2007, pp. 81–86.

[19] T. Sauter, "The continuing evolution of integration in manufacturing automation," *IEEE Industrial Electronics Magazine*, vol. 1, no. 1, pp. 10–19, 2007.

[20] M. Wollschlaeger, T. Sauter, and J. Jasperneite, "The future of industrial communication: Automation networks in the era of the internet of things and industry 4.0," *IEEE Industrial Electronics Magazine*, vol. 11, no. 1, pp. 17–27, 2017.

[21] H. Farhangi, "The path of the smart grid," *IEEE power and energy magazine*, vol. 8, no. 1, 2010.

[22] T. Sauter and M. Lobashov, "End-to-end communication architecture for smart grids," *IEEE Transactions on Industrial Electronics*, vol. 58, no. 4, pp. 1218–1228, 2011.

[23] M. Liserre, T. Sauter, and J. Y. Hung, "Future energy systems: Integrating renewable energy sources into the smart power grid through industrial electronics," *IEEE Industrial Electronics Magazine*, vol. 4, no. 1, pp. 18–37, 2010.

[24] C. Greer, D. A. Wollman, D. E. Prochaska, P. A. Boynton, J. A. Mazer, C. T. Nguyen, G. J. FitzPatrick, T. L. Nelson, G. H. Koepke, A. R. Hefner Jr *et al.*, "NIST framework and roadmap for smart grid interoperability standards, release 3.0," 2014.

[25] M. Schleipen, *Praxishandbuch OPC UA: Grundlagen - Implementierung - Nachrüstung - Praxisbeispiele*. Vogel Business Media, 2017.

[26] A. Dengel, *Semantische Technologien: Grundlagen. Konzepte. Anwendungen.* Spektrum Akademischer Verlag, 2011.

[27] F. van Harmelen, V. Lifschitz, and B. Porter, *Handbook of Knowledge Representation*, ser. Foundations of Artificial Intelligence. Elsevier Science, 2008.

[28] "Information and documentation – Thesauri and interoperability with other vocabularies – Part 1: Thesauri for information retrieval," ISO 25964-1:2011.

[29] "Information technology – Topic Maps – Part 2: Data model," ISO/IEC 13250-2:2006.

[30] "Topic Maps - XML Syntax," ISO/IEC JTC 1/SC 34, 2006, final Draft International Standard.

[31] "Unified Modeling Language 2.5.1," https://www.omg.org/spec/UML/2.5.1, 2017, last visited: Mar 2018.

[32] "OPC UA Specification, Release 1.03," OPC Foundation, 2015.

[33] "OBIX 1.1 Committe Specification," OASIS, 2015.

[34] "XSL Transformations (XSLT)," W3C Recommendation, 1999, last visited: Mar 2018. [Online]. Available: https://www.w3.org/TR/xslt

[35] "W3C XML Schema Definition Language (XSD) 1.1," W3C Recommendation, 2012.

[36] J. Tremblay and G. Cheston, *Data Structures and Software Development in an Object-oriented Domain*, ser. An Alan R. Apt Book Series. Prentice Hall/Pearson Education, 2003.

[37] W. Mahnke, S.-H. Leitner, and M. Damm, *OPC Unified Architecture*. Springer, 2009.

[38] R. Zurawski, *Industrial Communication Technology Handbook, Second Edition*, ser. Industrial Information Technology. Taylor & Francis, 2014.

[39] "RFC 7252 - The Constrained Application Protocol (CoAP)," 1999, last visited: Oct 2017. [Online]. Available: https://tools.ietf.org/html/rfc7252

[40] "Information technology – Common Logic (CL): a framework for a family of logic-based languages," ISO/IEC 24707:2007, 2007.

[41] R. J. Brachman and J. G. Schmolze, "An overview of the KL-ONE knowledge representation system," in *Readings in Artificial Intelligence and Databases*. Elsevier, 1988, pp. 207–230.

[42] F. Baader, *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2003.

[43] T. Berners-Lee and M. Fischetti, *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by Its Inventor*. Paw Prints, 2008.

[44] "OWL2 Web Ontology Language (Second Edition)," W3C Recommendation, 2012, last visited: Mar 2018. [Online]. Available: https://www.w3.org/TR/owl2-overview/

[45] A. De Nicola and M. Missikoff, "A lightweight methodology for rapid ontology engineering," *Communications of the ACM*, vol. 59, no. 3, pp. 79–86, 2016.

[46] "SPARQL 1.1 Overview," W3C Recommendation, 2013, last visited: Mar 2018. [Online]. Available: https://www.w3.org/TR/2013/REC-sparql11-overview-20130321/

[47] S. Villata, L. Costabello, N. Delaforge, and F. Gandon, "A Social Semantic Web Access Control Model," *Journal on Data Semantics*, vol. 2, no. 1, pp. 21–36, 2012.

[48] M. Grüninger and M. S. Fox, "Methodology for the Design and Evaluation of Ontologies," in *International Joint Conference on Artificial Intellienge, Workshop on Basic Ontological Issues in Knowledge Sharing*, 1995.

[49] M. Davis and H. Putnam, "A Computing Procedure for Quantification Theory," *J. ACM*, vol. 7, no. 3, pp. 201–215, Jul. 1960.

[50] L. Zhang and S. Malik, "The Quest for Efficient Boolean Satisfiability Solvers," in *Computer Aided Verification*, E. Brinksma and K. G. Larsen, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 17–36.

[51] E. Goldberg and Y. Novikov, "BerkMin: A fast and robust SAT-solver," *Discrete Applied Mathematics*, vol. 155, no. 12, pp. 1549–1561, 2007.

[52] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz, "Pellet: A practical owl-dl reasoner," *Web Semantics: science, services and agents on the World Wide Web*, vol. 5, no. 2, pp. 51–53, 2007.

[53] R. Shearer, B. Motik, and I. Horrocks, "HermiT: A Highly-Efficient OWL Reasoner." in *OWLED*, vol. 432, 2008, p. 91.

[54] R. Cupek and A. Maka, "OPC UA for vertical communication in logistic informatics systems," in *IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*, Sep. 2010, pp. 1–4.

[55] J. Virta, I. Seilonen, A. Tuomi, and K. Koskinen, "SOA-based integration for batch process management with OPC UA and ISA-88/95," in *IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*, Sep. 2010, pp. 1–8.

[56] "BACnet – A Data Communication Protocol for Building Automation and Control Networks," ANSI/ASHRAE 135, 2016.

[57] "OPC Unified Architecture," IEC 62541, 2012.

[58] "RFC 2616 - Hypertext Transfer Protocol – HTTP/1.1," 1999, last visited: Oct 2017. [Online]. Available: https://tools.ietf.org/html/rfc2616

[59] "SOAP Version 1.2," 2007. [Online]. Available: www.w3.org/TR/soap

[60] "RFC 6455 - The Websocket Protocol," 2011, last visited: Oct 2017. [Online]. Available: https://tools.ietf.org/html/rfc6455

[61] "RFC 5246 - The Transport Layer Security (TLS) Protocol Version 1.2," 2008, last visited: Oct 2017. [Online]. Available: https://tools.ietf.org/html/rfc5246

[62] "RFC 5246 - Datagram Transport Layer Security Version 1.2," 2012, last visited: Oct. 2017. [Online]. Available: https://tools.ietf.org/html/rfc6347

[63] "eXtensible Access Control Markup Language (XACML) Version 3.0," 2013, last visited: Oct 2017. [Online]. Available: https://www.oasis-open.org/standards#xacmlv3.0

[64] M. Jung, J. Weidinger, C. Reinisch, W. Kastner, C. Crettaz, A. Olivieri, and Y. Bocchi, "A transparent IPv6 multi-protocol gateway to integrate Building Automation Systems in the Internet of Things," in *IEEE International Conference on Green Computing and Communications (GreenCom)*. IEEE, 2012, pp. 225–233.

[65] T. Berners-Lee, J. Hendler, O. Lassila *et al.*, "The Semantic Web," *Scientific american*, vol. 284, no. 5, pp. 28–37, 2001.

[66] S. Bratt, "Semantic Web, and Other W3C Technologies to Watch," 2007, last visited: Oct 2017. [Online]. Available: https://www.w3.org/2007/Talks/0130-sb-W3CTechSemWeb/0130-sb-W3CTechSemWeb.pdf

[67] "RDF 1.1 specification," W3C Recommendations, 2014, last visited: Mar 2018. [Online]. Available: https://www.w3.org/RDF/

[68] "The JavaScript Object Notation (JSON) Data Interchange Format," 2014, last visited: Oct 2017. [Online]. Available: https://tools.ietf.org/html/rfc7159

[69] "The JSON Data Interchange Format, Standard ECMA-404," 2013, last visited: Oct 2017. [Online]. Available: http://www.ecma-international.org/publications/standards/Ecma-404.htm

[70] "RDF Schema 1.1," W3C Recommendation, 2014, last visited: Mar 2018. [Online]. Available: http://www.w3.org/TR/rdf-schema/

[71] "OWL 2 Web Ontology Language Mapping to RDF Graphs (Second Edition)," 2012, last visited: Oct 2017. [Online]. Available: https://www.w3.org/TR/2012/REC-owl2-mapping-to-rdf-20121211

[72] N. Nilsson, *Artificial Intelligence: A New Synthesis*. Elsevier Science, 1998.

[73] Z. Zhang and C. Zhang, *Agent-Based Hybrid Intelligent Systems: An Agent-Based Framework for Complex Problem Solving*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2004.

[74] "Building automation and control systems (BACS) - Room control functions (RA functions)," VDI 3813 Part 2, 2011.

[75] "Representation of process control engineering - Requests in P&I diagrams and data exchange between P&ID tools and PCE-CAE tools," IEC 62424:2016, 2016.

[76] C. Eastman, P. Teicholz, R. Sacks, and K. Liston, *BIM-Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers and Contractors*. Wiley John + Sons, 2011.

[77] "Industry Foundation Classes (IFC) for data sharing in the construction and facility management industries," ISO 16739:2016, 2016.

[78] "Open Green Building XML Schema: a Building Information Modelling Soltution for Our Green World," 2016, last visited: Mar 2018. [Online]. Available: www.gbXML.org

[79] "IEC 81346-1:2009 - Industrial systems, installations and equipment and industrial products - Structuring principles and reference designations - Part 1: Basic rules," 2009.

[80] "ISO/TS 81346-3:2012 - Industrial systems, installations and equipment and industrial products - Structuring principles and reference designations - Part 3: Application rules for a reference designation system," 2012.

[81] D. Grossmann, K. Bender, and B. Danzer, "OPC UA based Field Device Integration," in *SICE Annual Conference, 2008*, Aug 2008, pp. 933–938.

[82] "FDI Technical Specification 1.0," 2014.

[83] "Batch Control - Part 1: Models and Terminology," IEC 61512-1:1997, 1997.

[84] "Enterprise-control system integration - Part 1: Models and terminology," IEC 62264-1:2013, 2013.

[85] "OPC UA for ISA-95 Common Object Model 1.00 Companion Specification," ISA and OPC Foundation, 2013.

[86] D. van der Linden, H. Mannaert, W. Kastner, V. Vanderputten, H. Peremans, and J. Verelst, "An OPC UA interface for an evolvable ISA88 control module," in *16th IEEE Conference on Emerging Technologies Factory Automation (ETFA)*, Sept 2011, pp. 1–9.

[87] "Programmable Controller - Part 3: Programming Languaes," IEC 61131-3:2013, 2013.

[88] C. Legat, C. Seitz, and B. Vogel-Heuser, "Unified sensor data provisioning with semantic technologies," in *16th IEEE Conference on Emerging Technologies & Factory Automation (ETFA)*. IEEE, 2011, pp. 1–8.

[89] M. Neugschwandtner, G. Neugschwandtner, and W. Kastner, "Web Services in Building Automation: Mapping KNX to oBIX," in *2007 5th IEEE International Conference on Industrial Informatics*, vol. 1, June 2007, pp. 87–92.

[90] M. Jung, J. Weidinger, D. Bunyai, C. Reinisch, W. Kastner, and A. Olivieri, "Demonstration of an IPv6 multi-protocol gateway for seamless integration of Building Automation Systems into Constrained RESTful Environments," in *IEEE International Conference on Internet of Things (IoT 2012)*, 2012.

[91] M. Jung, J. Weidinger, W. Kastner, and A. Olivieri, "Heterogeneous device interaction using an IPv6 enabled service-oriented architecture for building automation systems," in *28th annual ACM symposium on applied computing*. ACM, 2013, pp. 1939–1941.

[92] ——, "Building Automation and Smart Cities: An Integration Approach Based on a Service-Oriented Architecture," in *2013 27th International Conference on Advanced Information Networking and Applications Workshops*, March 2013, pp. 1361–1367.

[93] M. Jung, J. Chelakal, J. Schober, W. Kastner, L. Zhou, and G. K. Nam, "IoTSyS: an integration middleware for the Internet of Things," in *4th International Conference on the Internet of Things (IoT 2014)*, Cambridge, MA, USA, Oct. 2014, demo abstract.

[94] "Internet Protocol, Version 6 (IPv6) Specification," RFC 2460, 1998, last visited March 2018. [Online]. Available: https://www.rfc-editor.org/rfc/rfc2460.txt

[95] "Efficient XML Interchange (EXI) Format 1.0 (Second Edition)," W3C Recommendation, 2014.

[96] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[97] K. Främling and M. Maharjan, "Standardized Communication Between Intelligent Products for the IoT*," *IFAC Proceedings Volumes*, vol. 46, no. 7, pp. 157 – 162, 2013, 11th IFAC Workshop on Intelligent Manufacturing Systems.

[98] R. Zach, H. Hofstätter, and A. Mahdavi, "A distributed and scalable approach to building monitoring," *eWork and eBusiness in Architecture, Engineering and Construction: ECPPM*, pp. 231–236, 2014.

[99] H. Dibowski and K. Kabitzsch, "Ontology-based device descriptions and device repository for building automation devices," *EURASIP Journal on Embedded Systems*, vol. 2011, no. 1, p. 623461, 2011.

[100] J. Ploennigs, B. Hensel, H. Dibowski, and K. Kabitzsch, "BASont-A modular, adaptive building automation system ontology," in *38th Annual Conference on IEEE Industrial Electronics Society (IECON)*. IEEE, 2012, pp. 4827–4833.

[101] H. Dibowski, O. Holub, and J. Rojícek, "Knowledge-based fault propagation in building automation systems," in *International Conference on Systems Informatics, Modelling and Simulation (SIMS)*. IEEE, 2016, pp. 124–132.

[102] J. Beetz, J. Van Leeuwen, and B. De Vries, "IfcOWL: A case of transforming EXPRESS schemas into ontologies," *Ai Edam*, vol. 23, no. 1, pp. 89–101, 2009.

[103] L. Daniele, F. den Hartog, and J. Roes, "Study on Semantic Assets for Smart Appliances Interoperability," European Commission, 2015.

[104] B. Balaji, A. Bhattacharya, G. Fierro, J. Gao, J. Gluck, D. Hong, A. Johansen, J. Koh, J. Ploennigs, Y. Agarwal *et al.*, "Brick: Towards a unified metadata schema for buildings," in *3rd ACM International Conference on Systems for Energy-Efficient Built Environments*. ACM, 2016, pp. 41–50.

[105] C. Reinisch, W. Granzer, F. Praus, and W. Kastner, "Integration of heterogeneous building automation systems using ontologies," in *Industrial Electronics, 2008. IECON 2008. 34th Annual Conference of IEEE*. IEEE, 2008, pp. 2736–2741.

[106] H. Wicaksono, S. Rogalski, and E. Kusnady, "Knowledge-based intelligent energy management using building automation system," in *International Power Electronics Conference (IPEC)*. IEEE, 2010, pp. 1140–1145.

[107] D. Bonino and F. Corno, "DogOnt - Ontology Modeling for Intelligent Domotic Environments," in *The Semantic Web - ISWC 2008*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 790–803.

[108] J. Han, Y.-K. Jeong, and I. Lee, "Efficient building energy management system based on ontology, inference rules, and simulation," in *2011 International Conference on Intelligent Building and Management, Singapore*, vol. 5, 2011, pp. 295–299.

[109] A. Mahdavi and M. Taheri, "An ontology for building monitoring," *Journal of Building Performance Simulation*, vol. 10, no. 5-6, pp. 499–508, 2017.

[110] A. Mahdavi, M. Taheri, M. Schuss, F. Tahmasebi, and S. Glawischnig, *Structured Building Data Management: Ontologies, Queries, and Platforms.* Cham: Springer International Publishing, 2018, pp. 261–286.

[111] J. Puttonen, A. Lobov, and J. L. M. Lastra, "Semantics-based composition of factory automation processes encapsulated by web services," *IEEE Transactions on industrial informatics*, vol. 9, no. 4, pp. 2349–2359, 2013.

[112] F. Rieckhof, H. Dibowski, and K. Kabitzsch, "Formal validation techniques for ontology-based device descriptions," in *16th IEEE Conference on Emerging Technologies & Factory Automation (ETFA).* IEEE, 2011, pp. 1–8.

[113] S. Fenz, J. Heurix, T. Neubauer, A. M. Tjoa, N. Ghiassi, U. Pont, and A. Mahdavi, "SEMERGY.net: automatically identifying and optimizing energy-efficient building designs," *Computer Science - Research and Development*, vol. 31, no. 3, pp. 135–140, Aug 2016.

[114] B. Vogel-Heuser, D. Witsch, and U. Katzke, "Automatic code generation from a UML model to IEC 61131-3 and system configuration tools," in *International Conference on Control and Automation (ICCA)*, vol. 2. IEEE, 2005, pp. 1034–1039.

[115] M. Schleipen, "OPC UA supporting the automated engineering of production monitoring and control systems," in *IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, Sept 2008, pp. 640–647.

[116] S. Runde and A. Fay, "A data exchange format for the engineering of building automation systems," in *IEEE International Conference on Emerging Technologies and Factory Automation (ETFA).* IEEE, 2008, pp. 303–310.

[117] V. Dubinin, V. Vyatkin, C.-W. Yang, and C. Pang, "Automatic generation of automation applications based on ontology transformations," in *IEEE Conference on Emerging Technology and Factory Automation (ETFA).* IEEE, 2014, pp. 1–4.

[118] "Function blocks - Part 1: Architecture," IEC 61499-1:2012, 2012.

[119] "SWRL: A Semantic Web Rule Language Combining OWL and RuleML," W3C Member Submission, 2004.

[120] H. Dibowski and K. Kabitzsch, "Semantic device descriptions based on standard semantic web technologies," in *IEEE International Workshop on Factory Communication Systems (WFCS).* IEEE, 2008, pp. 395–404.

[121] H. Dibowski, "Semantic interoperability evaluation model for devices in automation systems," in *Emerging Technologies and Factory Automation (ETFA), 2017 22nd IEEE International Conference on.* IEEE, 2017, pp. 1–6.

[122] S. Runde, H. Dibowski, A. Fay, and K. Kabitzsch, "Integrated automated design approach for building automation systems," in *IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 2008, pp. 1488–1495.

[123] S. Runde, A. Fay, and W.-O. Wutzke, "Knowledge-based Requirement-Engineering of building automation systems by means of Semantic Web technologies," in *7th IEEE International Conference on Industrial Informatics (INDIN)*. IEEE, 2009, pp. 267–272.

[124] S. Runde, H. Dibowski, A. Fay, and K. Kabitzsch, "A semantic requirement ontology for the engineering of building automation systems by means of OWL," in *2009 IEEE Conference on Emerging Technologies & Factory Automation (ETFA)*. IEEE, 2009, pp. 1–8.

[125] A. C. Oezluek, H. Dibowski, and K. Kabitzsch, "Automated design of room automation systems by using an evolutionary optimization method," in *IEEE Conference on Emerging Technologies & Factory Automation (ETFA)*. IEEE, 2009, pp. 1–8.

[126] H. Dibowski, J. Ploennigs, and K. Kabitzsch, "Automated design of building automation systems," *IEEE Transactions on Industrial Electronics*, vol. 57, no. 11, pp. 3606–3613, 2010.

[127] H. Dibowski, C. Oezluek, J. Ploennigs, and K. Kabitzsch, "Realizing the automated design of building automation systems," in *IEEE International Conference on Industrial Informatics (INDIN)*. IEEE, 2006, pp. 251–256.

[128] K. Kotis and A. Katasonov, "An ontology for the automated deployment of applications in heterogeneous IoT environments," *Semantic Web Journal (SWJ)*, 2012.

[129] "Semantic Sensor Network Ontology," W3C Recommendation, 2017.

[130] D. Retkowitz and M. Pienkos, "Ontology-based configuration of adaptive smart homes," in *7th workshop on Reflective and adaptive middleware*. ACM, 2008, pp. 11–16.

[131] M. Ruta, F. Scioscia, G. Loseto, and E. Di Sciascio, "Semantic-based resource discovery and orchestration in home and building automation: A multi-agent approach," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 1, pp. 730–741, 2014.

[132] A. Schumann, J. Ploennigs, and B. Gorman, "Towards automating the deployment of energy saving approaches in buildings," in *1st ACM Conference on Embedded Systems for Energy-Efficient Buildings*. ACM, 2014, pp. 164–167.

[133] M. Simros, S. Mätzler, S. Theurich, and M. Wollschlaeger, "Tool based device development method for rapid prototyping of profile conform field devices," in *IEEE Conference on Emerging Technology and Factory Automation (ETFA)*. IEEE, 2014, pp. 1–8.

[134] H. Dibowski, J. Ploennigs, and M. Wollschlaeger, "Semantic device and system modeling for automation systems and sensor networks," *IEEE Transactions on Industrial Informatics*, 2018.

[135] "Communication system for and remote reading of meters," EN 13757-2, EN 13757-3, 2016.

[136] "Communication systems for meters - Part 3: Application protocols," DIN EN 13757-3:2016, 2016.

[137] V. Gungor, D. Sahin, T. Kocak, S. Ergut, C. Buccella, C. Cecati, and G. Hancke, "Smart Grid Technologies: Communication Technologies and Standards," *IEEE Transactions on Industrial Informatics*, vol. 7, no. 4, pp. 529–539, Nov 2011.

[138] W. Kastner, F. Praus, G. Neugschwandtner, and W. Granzer, "KNX," *The Industrial Communications Handbook - Industrial Communication Systems*, pp. 42–1 – 42–14, 2001.

[139] "IEEE Standard for Ethernet," 802.3-2015, 2015.

[140] "Local Area Network: Token Bus," ATA 878.1 - 1999, 1999.

[141] "ZigBee 2015," ZigBee Aliance, 2015.

[142] T. Sauter and M. Lobashov, "End-to-End Communication Architecture for Smart Grids," *IEEE Transactions on Industrial Electronics*, vol. 58, no. 4, pp. 1218–1228, April 2011.

[143] M. Huq and S. Islam, "Home Area Network technology assessment for demand response in smart grid environment," in *Universities Power Engineering Conference (AUPEC), 2010 20th Australasian*, Dec 2010, pp. 1–6.

[144] Y. Strengers, "Smart Metering Demand Management Programs: Challenging the Comfort and Cleanliness Habitus of Households," in *20th Australasian Conference on Computer-Human Interaction: Designing for Habitus and Habitat*, ser. OZCHI '08. New York, NY, USA: ACM, 2008, pp. 9–16.

[145] "OPC Unified Architecture for Devices, Release 1.01," OPC Foundation, 2013.

[146] "OPC Unified Architecture Specification Part 8: Data Access, Release 1.03 ," OPC Foundation, 2015.

[147] W. Granzer, W. Kastner, and P. Furtak, "KNX and OPC UA," in *Konnex Scientific Conference*, Nov. 2010.

[148] "XML schema definition for OPC UA information models," last visited: Mar 2018. [Online]. Available: https://opcfoundation.org/UA/schemas/1.03/Opc.Ua.Types.xsd

[149] S. Mätzler, M. Wollschlaeger, A. Fernbach, W. Kastner, and M. Huschke, "An OPC UA cross-domain information model for energy management in automation systems," in *Industrial Electronics Society, IECON 2013 - 39th Annual Conference of the IEEE*, Nov 2013, pp. 7513–7518.

[150] J. A. Momoh, "Smart grid design for efficient and flexible power networks operation and control," in *Power Systems Conference and Exposition (PSCE'09). IEEE/PES*. IEEE, 2009, pp. 1–8.

[151] A. Fernbach and W. Kastner, "Integration of Smart Meters into Management Systems in Automation," in *10th IEEE International Workshop on Factory Communication Systems (WFCS)*, May 2014.

[152] S. Ashok, "Peak-load management in steel plants," *Applied energy*, vol. 83, no. 5, pp. 413–424, 2006.

[153] "ISO 16484 - Building automation and control systems (BACS) – Part 3: Functions," 2005.

[154] "OPC UA Information Model for IEC 61131-3, Release 1.00," PLCopen and OPC Foundation, 2010.

[155] "OPC Unified Architecture Specification Part 13: Aggregates, Release 1.03 ," OPC Foundation, 2015.

[156] M. Kofler, *An ontology as shared vocabulary for distributed intelligence in smart homes*. Vienna, Austria: PhD Thesis, Vienna University of Technology, 2013.

[157] M. Grassi, M. Nucci, and F. Piazza, "Ontologies for smart homes and energy management: An implementation-driven survey," *2013 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES)*, pp. 7–9, 2013.

[158] A. Fernbach, W. Granzer, and W. Kastner, "Interoperability at the Management Level of Building Automation Systems: A Case Study for BACnet and OPC UA," *16th IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*, Septempber 2011.

[159] C. Bizer, T. Heath, and T. Berners-Lee, "Linked data: The Story So Far," *Semantic Services, Interoperability and Web Applications: Emerging Concepts*, pp. 205–227, 2009.

[160] A. Fernbach, W. Granzer, and W. Kastner, "Knowledge Based Building and Energy Management," in *Tagungsband der e-nova 2015, Forschungs- und Studienzentrum Pinkafeld*, Nov. 2015.

[161] A. Fernbach, I. Pelesic, and W. Kastner, "Linked Data for Building Management," in *42nd Annual Conference of the IEEE Industrial Electronics Society (IECON)*, Oct. 2016.

[162] M. Kofler, "An ontology as shared vocabulary for distributed intelligence in smart homes," Ph.D. dissertation, Vienna University of Technology, 2013.

[163] "HTTP Over TLS," RFC 2818, 2000, last visited March 2018. [Online]. Available: https://www.rfc-editor.org/rfc/rfc2818.txt

[164] H. Dibowski, U. Ryssel, and K. Kabitzsch, "Ganzheitlicher, automatischer Entwurf drahtloser Gebäudeautomationssysteme," *at - Automatisierungstechnik*, vol. 61, 2013.

[165] "Construction drawings - Designation systems - Part 1 : Buildings and parts of buildings (ISO 4157-1 : 1998)," EN ISO 4157-1, 1998.

[166] "Information technology – Open Systems Interconnection – The Directory – Part 8: Public-key and attribute certificate frameworks," ISO/IEC 9594-8:2017, 2017.

[167] I. Ristic, *Bulletproof SSL and TLS: Understanding and Deploying SSL/TLS and PKI to Secure Servers and Web Applications*, ser. Computers / Security. Feisty Duck, 2013.

[168] J. Davies, *Implementing SSL / TLS Using Cryptography and PKI*, ser. IT Pro. Wiley, 2011.

[169] R. Oppliger, *SSL and TLS: Theory and Practice, Second Edition*, ser. Artech House information security and privacy series. Artech House Publishers, 2016.

[170] "Using OpenPGP Keys for Transport Layer Security (TLS) Authentication," RFC 6091, 2011, last visited March 2018. [Online]. Available: https://www.rfc-editor.org/rfc/rfc6091.txt

[171] N. Ferguson and B. Schneier. John Wiley and Sons, 2003.

[172] "Lightweight Directory Access Protocol," RFC 4511, 2006, last visited: Mar 2018. [Online]. Available: https://www.rfc-editor.org/rfc/rfc4511.txt

[173] C. Adams and S. Lloyd, *Understanding PKI: Concepts, Standards, and Deployment Considerations, Second Edition.* Addison-Wesley Professional, 2002.

[174] "Online Certificate Status Protocol," RFC 2560, 1999, last visited: Mar 2018. [Online]. Available: https://www.rfc-editor.org/rfc/rfc2560.txt

[175] O. F. Randy Armstrong and Y. Paul Hunkar, "The OPC UA Security Model For Administrators," *OPC Foundation*, July 2010.

[176] Microsoft Corporation, "Best Practices for Implementing a Microsoft Windows Server 2003 Public Key Infrastructure, www.microsoft.com/download/en/confirmation. aspx?id=20677," *Microsoft TechNet*, 2005, last visited: Mar 2018.

[177] "Working with Certificates," 2017, last visited: Mar 2018. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/framework/wcf/feature-details/working-with-certificates

[178] H. Sorenson, "An Introduction to OpenSSL, Part Three: PKI - Public Key Infrastructure," last visited: Mar 2018. [Online]. Available: https://www.symantec.com/connect/articles/introduction-openssl-part-three-pki-public-key-infrastructure

[179] "The OpenXPKI Project," last visited: Mar 2018. [Online]. Available: www.openxpki.org

[180] "Public-Key Infrastructure (X.509) (PKIX)," last visited: Mar 2018. [Online]. Available: datatracker.ietf.org/wg/pkix/charter

[181] "Java Security Overview," last visited: Mar 2018. [Online]. Available: https://docs.oracle.com/javase/9/security/java-security-overview1.htm

[182] "PKCS #10: Certification Request Syntax Specification Version 1.7," RFC 2986, 2000, last visited: Mar 2018. [Online]. Available: https://www.rfc-editor.org/rfc/rfc2986.txt

[183] W. Jansen, "Directions in Security Metrics Research," *NISTIR 7564*, April 2009.

[184] D. Schachinger and W. Kastner, "Integration von KNX Netzwerken in das Internet der Dinge: Die KNX Web Services Spezifikation," in *Tagungsband des 8. Jahreskolloquium "Kommunikation in der Automation - KommA"*, Dec 2016.

[185] C. A. Henson, H. Neuhaus, A. P. Sheth, K. Thirunarayan, and R. Buyya, "An ontological representation of time series observations on the Semantic Sensor Web," *The Ohio Center of Excellence in Knowledge-Enabled Computing (Kno.e.sis)*, 2009.

[186] N. Sharef, "Natural Language Query Translation for Semantic Search," *International Journal of Digital Content Technology and its Applications*, vol. 7, pp. 53–63, 10 2013.

Dipl.-Ing. Andreas Fernbach

TU Wien
Institute of Computer Engineering
Treitlstraße 1-3, A-1040 Wien

EDUCATION

Current studies: Doctoral programme of technical sciences
TU Wien — since March 2013

Doctoral College "Environmental Informatics"
TU Wien — March 2013 - February 2016

Master studies "Computer Engineering"
TU Wien — Graduation in February 2013

PROFESSIONAL EXPERIENCE

TU Wien — since October 2010
Research Assistant at the Institute of Computer Engineering

Research projects:

- Kognitive Regelstrategieoptimierung zur Energieeffizienzsteigerung in Gebäuden - FFG Energieforschung 1. Ausschreibung P848805
- Secure and Semantic Web of Automation (SeWoA) - FFG IKT der Zukunft P8040206
- OPC UA based Communication for Flexible Automated Manufacturing Cells (OPC4Factory) - FFG Produktion der Zukunft P843613
- Information Modelling in Automation (iModelA) - FFG EraSME/COIN P199 844
- Development of an IP multiplexing solution - FFG Innovations Cheque
- Webbased Communication in Automation (WebCom) - FFG EraSME/COIN P824 675

FH OÖ Campus Hagenberg — March 2013 - June 2013
Teacher in the master curriculum "Mobile Computing"

PUBLICATIONS

G. Zucker, A. Sporr, S. Kollmann, A. Wendt, L. S. Chaido, and A. Fernbach. A Cognitive System Architecture for Building Energy Management. *IEEE Transactions on Industrial Informatics*, 2018.

D. Schachinger, A. Fernbach, and W. Kastner. Modeling framework for IoT integration of building automation systems. *at - Automatisierungstechnik*, pages 630–640, 2017.

A. Fernbach and W. Kastner. Semi-automated Engineering in Building Automation Systems and Management Integration. In *Proceedings of the 26th IEEE International Symposium on Industrial Electronics (ISIE)*, pages 1528–1534. IEEE, 2017.

A. Fernbach and W. Kastner. Gebäudemanagement durch Wissensbasierte Systeme. In *Kommunikation und Bildverarbeitung in der Automation: Ausgewählte Beiträge der Jahreskolloquien KommA und BVAu 2016 Zum 10jährigen Jubiläum des InIT - Institut Für Industrielle Informationstechnik*, Technologien Für Die Intelligente Automation, pages 97–106. Springer Vieweg, 2017.

P. Hausberger, A. Fernbach, and W. Kastner. IMU-based smart fitness devices for weight training. In *Proceedings of the 42nd Annual Conference of the IEEE Industrial Electronics Society (IECON2016)*, pages 5182–5189, Oct 2016.

A. Fernbach, I. Pelesic, and W. Kastner. Linked Data for Building Management. In *Proceedings of the 42nd Annual Conference of the IEEE Industrial Electronics Society (IECON 2016)*, pages 6943 - 6945, 2016.

A. Fernbach and W. Kastner. Gebäudemanagement durch Wissensbasierte Systeme. In *7. Jahreskolloquium Kommunikation in der Automation - KommA*, 2016.

T. Frühwirth, F. Pauker, A. Fernbach, I. Ayatollahi, W. Kastner, and B. Kittl. Guarded state machines in OPC UA. In *Proceedings of the 41st Annual Conference of the IEEE Industrial Electronics Society (IECON 2015)*, pages 4187–4192, Yokohama, Japan, 2015.

A. Fernbach, W. Granzer, and W. Kastner. Knowledge Based Building and Energy Management. In *Tagungsband der e-nova 2015, Forschungs- und Studienzentrum Pinkafeld*, 2015.

A. Fernbach and W. Kastner. Integration of Smart Meters into Management Systems in Automation. In *Proceedings of the 10th IEEE International Workshop on Factory Communication Systems (WFCS 2014)*, 2014.

S. Mätzler, M. Wollschlaeger, A. Fernbach, and W. Kastner. Domänenübergreifende vertikale Integration mit OPC UA. In *Kommunikation in der Automation (KommA 2014)*, 2014.

W. Kastner, L. Krammer, and A. Fernbach. State of the art in Smart Homes and Buildings. In *Industrial Communication Technology Handbook, Second Edition*, pages 55–1–55–20. CRC Press, 2014.

A. Fernbach, W. Kastner, S. Mätzler, and M. Wollschlaeger. An OPC UA Information Model for Cross-Domain Vertical Integration in Automation Systems. In *Proceedings of the 19th IEEE Conference on Emerging Technologies and Factory Automation (ETFA'14)*, 2014.

W. Kastner, A. Fernbach, W. Granzer, and M. Jung. KNX and the Semantic Web of Automation. In *Proceedings of the KNX Scientific Conference*, 2014.

S. Mätzler, M. Wollschlaeger, A. Fernbach, and W. Kastner. An OPC UA cross-domain Information Model for Energy Management in Automation Systems. In *Proceedings of the 39th Annual Conference of the IEEE Industrial Electronics Society (IECON 2013)*, pages 7513 - 7518, 2013.

A. Fernbach, W. Granzer, W. Kastner, and P. Furtak. Mapping ETS4 Project Structure to OPC UA using ETS4 XML Export. In *Proceedings of the KNX Scientific Conference*, 2012.

A. Fernbach and W. Kastner. Certificate Management in OPC UA Applications: An Evaluation of different Trust Models. In *Proceedings of the 17th IEEE Conference on Emerging Technologies and Factory Automation (ETFA 2012)*, 2012.

A. Fernbach and W. Kastner. Information modelling in OPC UA. In *Tagungsband SPS/IPC/Drives*, 2012.

A. Fernbach, W. Granzer, and W. Kastner. Interoperability at the Management Level of Building Automation Systems: A Case Study for BACnet and OPC UA. *Proceedings of the 16th IEEE Conference on Emerging Technologies and Factory Automation (ETFA 2011)*, Septempber 2011.

G. Neugschwandtner and A. Fernbach. Design of an enhanced TP-UART based KNX PC Interface. In *Proceedings of the Konnex Scientific Conference*. KNX Association, 2008.