White Paper

Brownfield Devices in IIoT

Automating the Integration via Semantic Technologies

fortiss

Brownfield Devices in Industrial Internet of Things –
Automating the Integration via Semantic Technologies

Authors

Kirill Dorofeev

fortiss GmbH
Guerickestr. 25
80805 Munich

Hendrik Walzel

fortiss GmbH
Guerickestr. 25
80805 Munich

Prof. Dr. Rute C. Sofia

fortiss GmbH
Guerickestr. 25
80805 Munich

kontakt@fortiss.org

Content

Abstract

This whitepaper provides an overview on semantic interoperability aspects within the context of Industrial Internet of Things (IIoT) by fortiss, specifically focusing on concepts that can assist a much needed automated integration of Operational Technology into Information Technology, from an end-to-end perspective. The whitepaper provides the vision of fortiss for such integration to happen, namely, aspects that need to be addressed from both an application/session layer and networking layer perspective. The whitepaper describes also a specific instantiation being under development by fortiss within the context of its Industrial IoT Lab, namely, the "IIoT Lab Brownfield device integration demonstrator".

1 Introduction

The recent trend of *Flexible Factory* [1] implies, at an ultimate level, the integration of sensors and other *Internet of Things (IoT)* devices and systems on industrial plants to improve monitoring, management as well as the overall operation of industrial elements, e.g., machines, robots. The data collected via IoT systems is used in several aspects. For instance, it can be used to perform local or remote maintenance, providing status data about a specific environment or machine. It can also be used for the fast replication of real assets via virtual representation (e.g., digital twin). Or, it can support a better development of services in real-time, tailored to the requirements and capabilities of different machines with the ultimate aim to improve efficiency and productivity.

In industrial environments, advanced IoT equipment today co-exists with a large number of legacy devices also coined as *brownfield devices*, i.e., devices which cannot, per se, be integrated into IoT platforms, be it Cloud- or Edge-based, given that the description of such devices is not understood by more recent IoT systems. Examples of brownfield devices are, for instance, industrial robots, or *Programmable Logic Controllers (PLCs)*.

Brownfield devices are part of the category of the so-called *Operational Technology (OT)*. OT equipment has a long lifetime often running for decades and implies heavy investment, thus OT replacement is not easy to do.

Moreover, due to the lack of automated data analytics support, OT is often managed via manual processes. OT data is often manually handled by a human operator. This manual process leads to mistakes and to the lack of a systematic, standardized data recording, often preventing also a comparison of performance across different devices thus preventing a higher degree of efficiency.

One approach to assist in integrating the OT data and eventually associated processes, is to consider reliable solutions that can automate the data collection of such devices locally (on the shop-floor). This means collecting and analyzing data from legacy equipment without adequate end-to-end infrastructure. Such approach is often based on proprietary hardware/software solutions which are tailored to specific equipment and to specific, usually proprietary, protocols for data exchange. Hence, the design of the underlying communication architecture often does not consider aspects such as security, since the exchange of data is done locally, in trusted environments. The devices operation and their maintenance are also performed locally.

The interconnection of brownfield devices into end-to-end IoT systems assists the development of novel services as well as to perform administration and maintenance remotely. Moreover, advanced data analytics and even advanced sensing methods can also be integrated into industrial environments. However, such integration faces several challenges. A first challenge concerns security. IoT protocols/protocol frameworks such as OPC-UA and MQTT have been developed to run on top of the TCP/IP stack. End-to-end security by design is not an intrinsic feature of current IP-based messaging protocols (e.g., MQTT, AMQP). OPC-UA as a protocol framework supports secure communication between data sources and consumers. However, OPC-UA cannot ensure the complete isolation of a plant network when connecting IIoT infrastructures, as to access data from OPC UA server, an OPC UA client outside the plant network needs an open firewall port. The answer currently provided to this issue is to keep OPC-UA for the in-plant communication, and to provide interoperability with other IP-based messaging approaches, for instance OPC-UA to MQTT, to support the articulation of communication to the overall end-to-end IoT system (Edge-Cloud).

A second challenge concerns the need to support asynchronous communication both at a field level, and from Edge to Cloud.

A third aspect concerns the need to provide a compatible "Thing Description", so that IoT systems can interpret brownfield devices as a *"Thing"*, or as a set of Things. To further assist a smoother integration of legacy devices, open-source end-to-end IIoT systems need to:

1. integrate middleware that can provide a meaningful software-based abstraction of brownfield devices;
2. provide a compatible semantic description of a brownfield device to an IoT system. In other words, such solution would have a way to interpret the aspects that compose the device (measurement, state, and device's functionality) as the aspects that compose a Thing.

On the field-level, there are a few solutions being worked upon to assist this integration. For instance, some devices which can be upgraded are being coupled with specific protocol, such as OPC-UA or MQTT Sparkplug bindings. The protocol bindings are the basis to send data to the IoT Edge/Cloud.

Another approach being considered concerns relying on a software-based solution (a soft gateway) that then is tailored to the specific support of legacy devices on a specific environment.

Our approach considers that a software-based abstraction can assist in defining a generic device not only in regards to its properties, but also in regards to its capabilities and functionalities. Such abstraction can then be fed to an IIoT system, and be interpreted as a regular, virtual "Thing", or set of "Things". Hence, a key part of this concept concerns the development of an automated software-based solution that receives as input a skill-based model of a brownfield device, and provides as output a semantic description of a Thing, based on *Web of Things (WoT)* standards.

This paper is therefore focused on a debate of concepts currently under development by fortiss, and which address how to best support the integration of brownfield devices in IIoT systems, from an end-to-end perspective.

The main contributions of this white paper are as follows:

- the paper provides a notion of semantic interoperability in the context of IIoT, and in particular, in the context of an automated integration of OT systems into IT systems.
- The paper describes concepts and technologies being developed at fortiss to further assist such integration, specifically focused on
  i) assisting a bi-directional communication to OT equipment, from an IIoT system;
  ii) concepts derived from intent-based networking, which may help in making the underlying infrastructure adjust, with a reduced need for human intervention, via the use of expressions of interest derived from services and applications, coined as "Intents".
- The paper summarizes research questions being addressed by fortiss in the context of semantic interoperability in IIoT environments.
- The paper describes the current demonstrator being developed at fortiss, in the IIoT Lab, highlighting novel software concepts under development.

The white paper is organized as follows. After this introduction, section II describes related literature, highlighting our expected contributions. Section III provides background on semantic interoperability aspects that are relevant to understand the context of our work, such as the role of the World Wide Web Consortium (W3C) standards; how different entities are handling interoperability of data models; a model for an extensible semantic gateway; the skill-based engineering model for abstracting brownfield devices status, processes and properties. Section IV focuses on new paradigms concerning the required IoT infrastructure adaptation, by focusing on "what" is to be processed, and not on "who" (host, device) is the data source, namely, debates Intent-based networking and the roles of Intents, and provides a brief description of Information-centric Networking (ICN) as a relevant architecture for future IIoT systems. Section V provides an overview of the research questions and directions being addressed in fortiss. Section VI describes the fortiss IIoT Lab demonstrator for brownfield device integration for which a first version shall be available in November 2020, while section VII summarizes the white paper and its contributions.

II Related Work

A first category of work related to ours, concerns the description of IoT cyber-physical systems *(Things)* and its communication interfaces in a universal way, avoiding silo deployments both in terms of equipment/software capabilities, and of data exchange capabilities. This is, for instance, the main aim of the W3C, which has recently adopted as official Recommendations the *Web of Things (WoT) Architecture*[1] and the *WoT Thing Description (WoT TD)*[2] [2] in the context of its WoT working group[3]. The WoT architecture recommendation provides an overall conceptual framework for a WoT, proposing specific building blocks to achieve end-to-end interoperability in IoT environments. The WoT TD specification provides a universal model for both meta-data and interfaces of Things and is further addressed in section III. Such approach assumes that the IoT devices can be modeled as WoT Things, i.e., that they can be expressed via a JSON-LD format.

Our work considers the WoT as a centerpiece, but assumes that some devices are not capable of expressing the meaning of their interactions, and, therefore, require an additional level of abstraction to be interpreted as Things, as described in section III-E. In this case, the intention is to use and eventually extend WoT specifications to smoothly incorporate properties derived from, e,g., a skill-based engineering model of field-level devices.

Interesting in this context are IoT integration platforms such as Wotify [3], a W3C intermediary platform, which allows developers to quickly create their projects by uploading or reusing WoT TD descriptions for sensors, actuators, or other types of equipment modeled via a WoT TD. Being highly relevant to increase the use of WoT, it does not contemplate an automated integration of legacy devices, an aspect which is pursued with this work.

Additional related work has been attempting to contribute to an automated integration of legacy devices in specific application domains. For instance, Mossamer et al. address such integration in regards to the energy domain [4]. The authors introduce an abstraction approach, an automated embedded annotation engine, that reduces the need to depend on specific addressing schemes and can handle devices supporting legacy automation protocols based on IEC 60870-5-104, Modbus. Such engine reduces the need to handle each legacy device individually (providing augmented annotation for each device). Rachetti et al. provide a proposal for an *Abstraction Layer Object Oriented Architecture (ALOOA) and its application to Motion Control* [5]. ALOOA is an Abstraction Layer for PLCs that relies on Object-oriented features from IEC 61131-3. In contrast, our skill-based model approach described in section III-E aims at being agnostic of the underlying implementation.

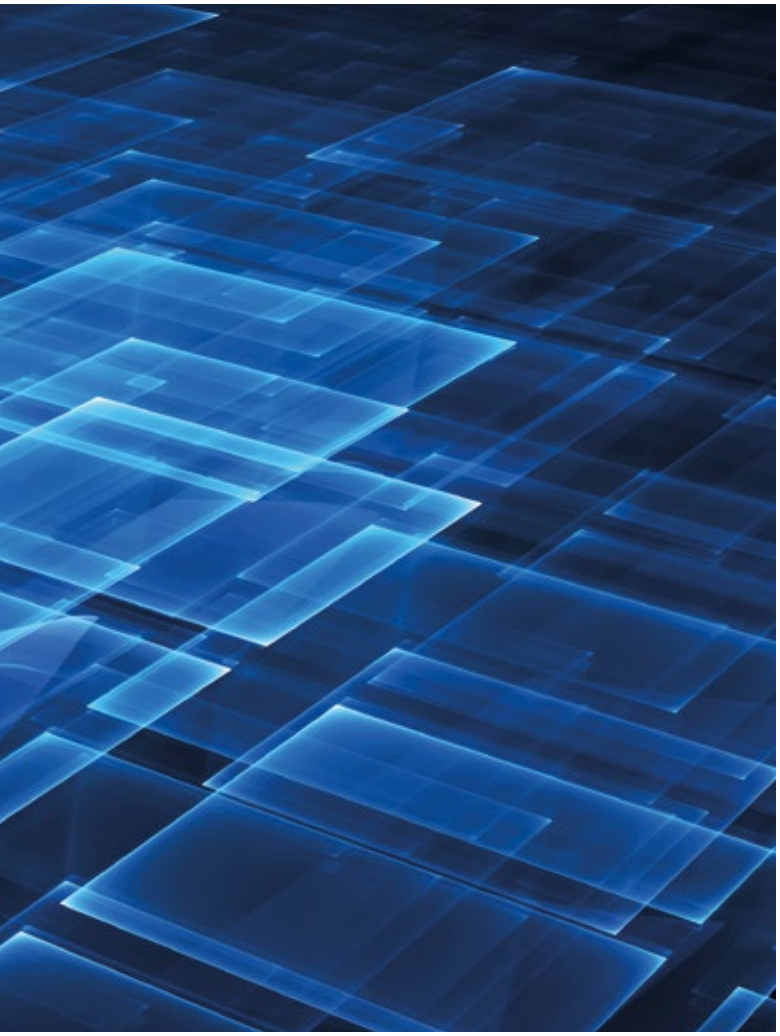
Petrolaukis et al. provide the notion of a semantic gateway and an end-to-end semantic architecture for the IIoT, specifically addressing the application domain of eHealth and Energy [6]. A middleware, the *Semantic Mediator*, assists in providing a semantic mediation abstraction for legacy devices, by mapping a specific domain brownfield semantic standard into WoT TD. Our work shall build on the Semantic Mediator component, as is further explained in sub-section III-C, and shall provide a mapping to another type of abstraction model, the skill-based engineering model, described in section III-E, in a way to prevent silos developed around specific protocols.

Another category of work related to ours concerns attempts to automate the description not only of measured values but also of the sequential behavior of devices. This is particularly relevant to assist an adequate integration of legacy equipment into end-to-end IIoT environments. In this category of work, Korkan et al. propose an extension to WoT TD which increases semantic expressiveness and provides a way to integrate valid state transitions [7]. According to the authors, behavior of devices can then be expressed so that devices can be handled as part of an IIoT ecosystem, thus reducing overall manual intervention. Our work shall consider the proposal of Korkan et al. and check the capability of their proposal to integrate the logic and state machine of PLCs abstracted via our skill-based engineering model.

A final category of related work concerns the need to also adapt the underlying infrastructure, i.e., to bring automation to the network interconnecting OT and IT, to best support service automation end-to-end. This line of related work, *Autonomic Networking* [8], looks into the adaptation of hardware-centric and manually managed networks into controller-led networks, that relies on business/service expressions of interest *(Intents)* [9] and translates such Intents into network policies that can then be automated so that the network devices can continuously monitor and adjust the network performance to assist in reaching the desired (business) goals. This notion, defined as *Intent-based*

1 https://www.w3.org/TR/wot-architecture/
2 https://www.w3.org/TR/wot-thing-description/
3 https://www.w3.org/WoT/WG

Networking (IBN) [9], is relevant also to be applied into the smooth integration of OT and IT. *Intents* in this context are abstract, high-level descriptions of how the services can be automatically implemented from an end-to-end perspective, providing the automated means to establish a network that best serves the respective service automation. These abstractions, therefore, model the network, by expressing *"what"* the end-user/field-level device needs from the network, without handling *"how"* it will be handled by the network. Our work shall address the definition and classification of Intents in the context of brownfield integration, attempting to come up with automated ways to define and classify Intents [10] and assist the network in adjusting to the needs of devices and of their processes.



III Semantic Interoperability Background

A full integration of individual devices into an overall system requires interoperability on three levels: *technical, syntactical* and *semantic* interoperability [11]. Technical interoperability refers to connectivity-related aspects and the respective hardware as well as software components that enable communication. Syntactical interoperability specifies data formats as well as communication protocols and defines the syntax as well as encoding mechanisms for data exchange. Semantic interoperability describes the meaning of the exchanged data and thereby creates a common understanding between all involved components [11]. In that sense, our approach aims at bridging the gap in regards to allowing legacy devices to be integrated into end-to-end IIoT systems, in a way that supports such devices as Things. This section goes over different approaches being applied to support the integration of brownfield devices into sophisticated, end-to-end IoT systems.

A. The Web of Things Approach

As described in section II, the W3C has recently released official recommendations for a WoT architecture and a WoT TD. The WoT provides a software-based, multi-domain and multi-network semantic interoperability layer. Its semantic vocabulary is aligned with *iotschema* (rf. to section III-B). Therefore, the WoT TD provides both a way to abstract Things and to support interfaces to the required IoT protocols. Therefore, WoT descriptions allow IoT data to to run over different protocols.

A WoT TD is seen as an entry point for a Thing, similarly to the function that an *"index.html"* has in a website. A WoT TD has 4 main components: textual meta-data related with different protocol bindings and identified by URI schemes; a set of *Interaction Affordances*, which provides a description for specific action and even triggered interactions (how the Thing is used); machine readable schemas for data exchange; Web links to express formal or informal relations to Things and to Web documents.

Interaction Affordances currently comprise three main categories: properties, actions and events. *Properties* model sensed/control values (including setting an operation state). *Actions* model the invocation of physical processes, e.g., to manipulate the internal state or start a process. They can also be used to abstract RPC-like calls. Events are used

to model a sender-driven, push-based, asynchronous communication, where notifications, discrete events, or streams of values are sent asynchronously to receivers.

It is relevant to highlight that a TD is quite flexible, and integrates the possibility to consider *contextual definitions* for a specific namespace. Context can therefore assist in modeling formal knowledge associated with a specific application domain. It can also provide a way to specify configuration and behavior associated to specific protocols (declared in the *forms* field).

B. Providing semantic definitions: iotschema.org

iotschema[4] is an open effort that provides shared vocabularies for IoT domains. It provides a common semantic layer for interconnecting Things based on different modeling languages. In IIoT, iotschema provides the tools required to create semantic definitions on specific domains. iotschema is used, for instance, by IoT platform providers to assist in a smoother integration of third-party applications. iotschema can also be used by device vendors to push for a Web wide adoption. Finally, it is also relevant to provide application portability in different environments, and to create domain-specific languages to support the interconnection of Things.

A highly relevant aspect of iotschema is the possibility to specify *capabilities* of objects in the form of interactions, thus, assisting objects of different domains to more easily interconnect. An iotschema capability is the smallest composeable unit of functionality and is used as a basis to compose interaction patterns.

The iotschema Meta-model is composed of the following elements:

- Property, readable and optionally writable state element.
- Action, a parameterized incoming state change with rich responses.
- Event, a parameterized outgoing state change; a message describing some occurrence.
- Capability, a set of properties, actions, events that provide common interaction affordances. It usually relates with a scoped function and is defined with some semantic meaning, e.g., "on/off", "temperature measurement". Can be used to support the definition of larger aggregations.
- Data types, corresponding to associate semantic meaning with data constrains, e.g., Temperature data; number type; units.

C. The SEMIoTICS Approach to Brownfield Integration

The SEMIoTICS project[5] is focused on the support of Industrial IoT ecosystems comprising sensors and field-level devices, for the purpose of a better monitoring.

A specific use-case of SEMIoTICS is based on the interaction between a legacy device and its controller (a wind turbine with a specific control system – Siemens SIMATIC S7). The measurements provided by sensors are obtained via an OPC-UA server, running on a Siemens SIMATIC PLC, i.e., the end-point[6]. So a question answered by the SEMIoTICS project is: *"how to integrate IoT sensors into existing automation systems"*. For that purpose, SEMIoTICS has contributed with specific semantic specifications of brownfield devices based on iotschema. The approach followed by SEMIoTICS is therefore one of a semantic mapping from brownfield semantic models into IoT semantic models.

Such semantic mapping aims at providing a unified support for semantic standards of different domains into iotschema. The SEMIoTICS gateway [5] addresses this unification in terms of semantic packs (or semantic nodes). Therefore, it aims at handling brownfield integration per domain.

Within the semantic gateway, a component, the *Semantic Mediator*, serves the semantic mapping between different data models [12]. The Mediator caters for greenfield and brownfield devices. The function of the Mediator is to make a device programmatically accessible based on the mapped WoT TD.

In the case of the brownfield devices being connected to the mediator, an additional step is required in order to create a WoT TD based on the information extracted from the brownfield device. In [6], Petroulakis et al. show a potential way of providing such a mapping from existing PLC into a WoT TD. After exporting the tags from a PLC program, the Mediator creates the corresponding programmable components, e.g., *Device Nodes* to allow for interaction with the device, based on their WoT TD. The Mediator then provides access to the device capabilities via a standardized Web API which runs on a WoT servient component[7].

4 https://iotschema.org/

5 https://www.semiotics-project.eu/

6 OPC-UA follows a client/server architecture where clients are more sophisticated than clients in the usual client/server architectures, and where servers reside in end-points.

7 https://www.w3.org/TR/wot-architecture/

D. Towards Unified Data Models, OneDM

The integration of individual devices into an overall system requires all hardware and software components to share the same meaning about the provided data and available functionalities. Within one domain, most devices serve the same context, purpose and objectives. Therefore, such a common understanding is implicitly given by knowledge about that domain, or documented in manuals or technical reports. Still, mechanisms that aim to integrate devices automatically require explicit, formally defined models.

Most legacy machines used in production contain legacy PLCs. Those PLCs provide a vendor-specific interface to access devices' data. Vendors, such as, for example, Siemens and Beckhoff, introduced proprietary fieldbus systems to enable data exchange between single controllers and thereby devices, i.e. machines and systems. Those fieldbus technologies comprise vendor-specific specifications regarding the hardware and software interface. Furthermore, they introduced individual models that describe the location as well as the meaning of the data that can be accessed. Thereby, the different fieldbus technologies created enclosed ecosystems with very limited interoperability functionalities. Therefore, the integration of devices from different ecosystems requires a lot of effort as well as further hardware equipment and software tools. One approach to facilitate the interoperability between ecosystems is the establishment of domain-wide standards and models.

Out of the different approaches that attempt to provide some level of unification in regards to data models, the recent (2019) *One Data Model (OneDM)* [8] effort should be accounted for. OneDM is a voluntary effort developed by several IoT *Standards Development Organizations (SDOs)* and IoT device and platform vendors. It focuses on creating consistency between the different IoT data models available via the use of the *Simple Definition Format (SDF)* for OneDM definitions. OneDM relies on a meta-model that is similar to iotschema (rf. to section III.B.), and SDF can be used to create iotschema definitions.

E. Abstracting Legacy Devices: a Skill-based Engineering Abstraction Model

Albeit automation systems are becoming smarter, via the introduction of sophisticated technologies, in industrial domains there is still a massive number of legacy devices that execute production and that are managed in a semi-automatic way. The integration of such devices with a low effort into smarter IIoT environments, being able to take advantage of their data to improve aspects such as reliability and efficiency are crucial for industrial domains. The centerpiece of almost every control system is a PLC, which is normally programmed to control a specific process by gathering the information from sensors and manipulating actuators. In this context, Dorofeev et al. have introduced a generalized skill interface. This interface represents the data exchanges and function invocations that constitute a system operation for control components [14][15].

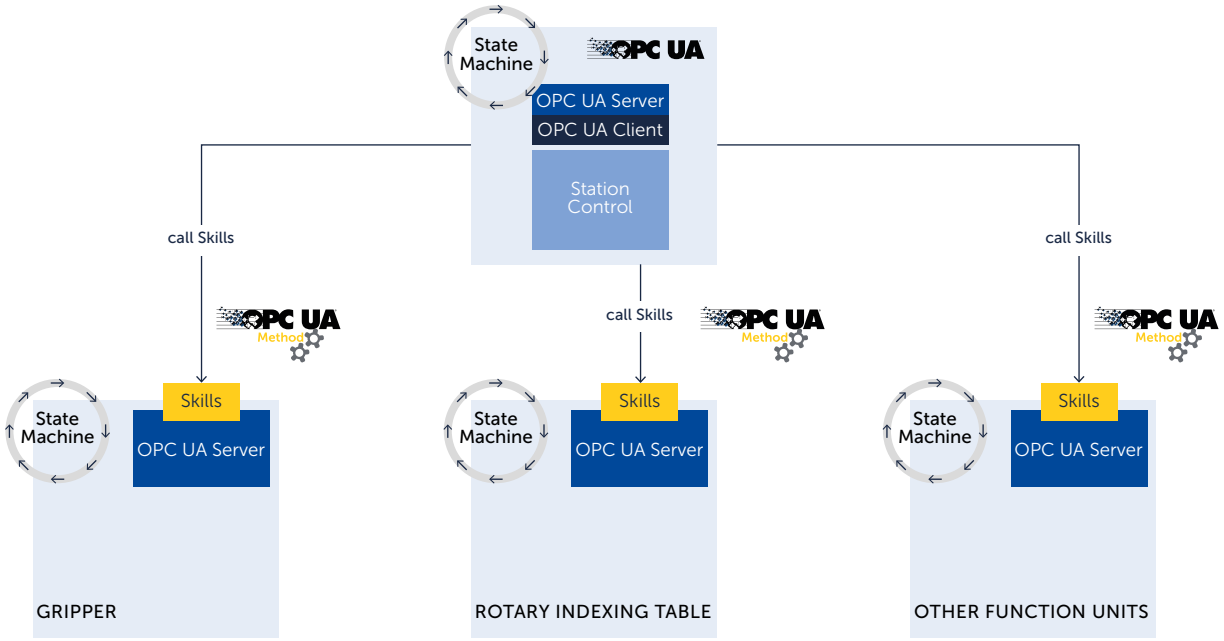


Figure 1: A skill concept [15].

8 https://onedm.org/, https://github.com/one-data-model/oneDM

A *skill* provides an abstraction interface to access the devices' functionality. A generic skill model remains the same for any functionality of a hardware or software module being offered by that module. Moreover, skills exist at different levels of a control hierarchy and can be combined, often following the hardware structure of a machine, as it is shown in Figure 1. The skill model itself is defined in a protocol-independent way. An instance of this model has been realized using the OPC-UA information model in its first reference implementation [14].

The skill model defines a set of states, control methods, and events that represent i) the behavior of a control component, ii) a way to trigger it and iii) the intermediate and resulting data for each execution. This is a flexible abstraction approach, that can be easily extended in order to reflect the specifics of a skill of any complexity. The concept of skill-based engineering deals with designing automation systems based on adequate orchestration of required skills needed for realizing a particular production process steps. In this way, manufacturers of automation components focus on developing and providing resources that offer certain functionalities that match the required skills for a certain process steps. That strategy allows new possibilities in terms of engineering and configuration of automation systems and increases cross-vendor interoperability.

The skill-based state machine, as illustrated in Figure 2, is composed of a minimum state machine with its methods and events. The generic skill-based state machine, describing the PLC's functionality, has four states, which are mandatory for each skill:

1. **Locked:** Starting the skill execution is not allowed and no physical motion or production data transformation takes place. An *Error* substate is usually reasonable.
2. **Idle:** Represents the availability of a skill for execution. No motion or data processing takes place.
3. **Executing:** The skill is being executed and motion or production data transformation can take place.
4. **Suspended:** Halted state which can be either recovered to *Executing*, in which the parameters passed with the original *start*() command are reused, or cleared to return to *Idle*.

Knowing these four general states of a skill, an overall orchestrator can control the execution of a skill, triggering the state changes (for example, from Idle to Executing) and observing the current state. Whenever a skill state machine is changing its state, a transition event is fired which can be monitored by the orchestrator, so that it can make further process steps. For example, in Figure 1 the overall cell orchestrator is controlling its components over such state machines, executing their skills in a sequence required for the production process. If needed, the four mandatory states can be extended by an indefinite number of substates, which makes the overall model flexible and capable of modeling a large variance of possible functionalities.

An OPC-UA server represents the skill as an object in its address space, offering a generic interface that enables an access to the PLC functionality in a universal way. This object can be of any type that inherits from the *SkillType*, an abstract type that serves as a parent class for all skill implementations. *SkillType* should define a capability that the respective skill fulfills. These capabilities can be classified in some ontology, for example, VDI 2860, that defines a set of handling operations and can be used for such classification of the individual production steps.

The skill object itself defines an instance – a specific implementation – of a skill with all the details that a potential skill consumer should know about it before executing the skill. This includes all of the states, control methods and transition events that this skill instance has as well as all additional data, such as input/output parameters, etc.

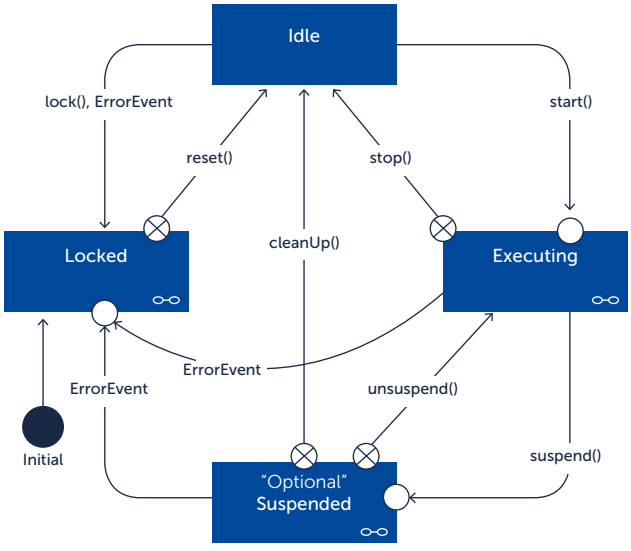


Figure 2: Skill-based state machine.

IV Interoperability at the Network Level

A. Adapting the Infrastructure: Intents and Intent-based Networking

Modern IoT applications are composed of several services that provide, share and consume data over networked environments, such as industrial plants [16]. The elements that compose such an IIoT environment are often mobile and communicate over heterogeneous networking protocols and different protocolar frameworks (and different protocol versions). The complexity to handle the required interconnection increases with the number of devices involved (sources and destinations), services, and technologies applied. In these scenarios, the network remains static, i.e., network management is often provided in a static way, and reconfiguration requires human intervention. This results in configuration errors and in environments that can hardly adjust to novel services. However, modern IoT applications such as AR/VR, AI, require a network that adapts to changing IoT application configurations and their demands.

One way to make the network adaptable is to consider the concept of Intents (rf. to section II), where Intents can be expressed via semantic technologies. Intents are applied in multiple environments. For instance, in autonomic networking and with the purpose of assisting self-management, Intents are user-defined policies, i.e., *"An abstract, high-level policy used to operate the network. Its scope is an autonomic domain, such as an enterprise network. It does not contain configuration or information for a specific node (...). It may contain information pertaining to a node with a specific role (for example, an edge switch) or a node running a specific function. Intent is typically defined and provided by a central entity"* [8]. It serves as an interface between the network and Intent Users, e.g. service operators, network administrators, or application developers, that request certain network functionalities. Intent Users declare what operations shall be performed by the network without providing functional or operational details. The network determines courses of actions and triggers functions for orchestration, configuration, monitoring, and measurement. Thereby, the network is able to adapt to current application demands. Furthermore, Intents are invariant to the network infrastructure[9]. This means that Intents do not change when the infrastructure changes, for instance, when nodes or links become unavailable, or when mobile devices roam. Monitoring and measurement functions inside the network evaluate its performance and compare them to the desired requirements given by the Intent.

Cisco applies the concept of Intent-based Networking to automate and therefore facilitate the management of enterprise networks. There, Intents are specified in business language and describe what the business wants from the network, e.g. *"I want these servers to be reachable from these branches; therefore, I need to configure specific VLAN, subnet, and security rules on each device in my network"*[10]. The respective low-level network policies that execute a given intent, the how, are generated within the Intent-based network. The network *"continuously monitors and adjusts network performance to assure the desired business outcome"*[11]. For this, Cisco defines three essential functions: *Translation*, *Activation*, and *Assurance*. Firstly, a given business intent is translated into network policies and checked for integrity. Secondly, the IBN orchestrates the generated policies and configures the low-level system components. Thirdly, the execution is permanently monitored and checked against the given business intent. If drifts occur, corrective actions are triggered autonomously.

In Intent-Driven Networks as proposed by Elkhatib et.al. [17], an Intent is defined as a tuple of a *verb, object, modifiers,* and *subject*. The verb, which expresses the desired operation, is specified using a given ontology and further parameterized by the modifiers. The object and the optional subject refer to services, objects, or items that are objectives of operation. Elhabbash et.al. use those Intent definitions to adaptively select service instances at application runtime [18]. There, two instances of an online document editing service, i.e. Google Docs, are deployed at different location within the network, i.e. at the Edge and on the Cloud. Different service user clients define and send *Quality of Service (QoS)* requirements, e.g., response time, that need to be fulfilled when they want to connect to the service. A mediator component inside the network receives those requirements and selects as well as connects the user clients to a specific service instance. When user clients connect or disconnect from the service, the mediator reevaluates the current set of received Intents and, if necessary, adapts established connections.

9 https://datatracker.ietf.org/meeting/interim-2019-nmrg-07/materials/slidesinterim-2019-nmrg-07-sessa-intent-based-network-summit-2015

10 https://www.cisco.com/c/dam/en/us/solutions/collateral/enterprisenetworks/ digital-network-architecture/nb-09-intent-networking-wp-cte-en.pdf

11 https://blogs.cisco.com/datacenter/how-to-create-networks-that-meetyour-intent

Intents can, however, have other forms. For instance, in LTE Direct Intents are defined as *expressions* [19] sent via LTE beacons to devices in the vicinity, to broadcast specific information. LTE Direct Expressions are 128-bit service layer identifiers that can represent different things, for instance, an object identity, a service, a capability, or location. Thereby, things are able to inject meta-data about its context, etc. to the system without further specification how these should be processed.

In the Android *Operating System (OS)*, Intents are defined as messaging objects that are *"used to request an action from another app component"* [12]. In case the other app component is unknown at design time, the developer creates an implicit Intent to describe the desired action, e.g. taking a photo. Such an Intent comprises

- an *Action* string that specifies a generic action to perform;
- e.g. ACTION_IMAGE_CAPTURE[13];
- a *Data* URI that references the data to be acted on and/or the MIME type of that data;
- a *Category* string that defines the kind of component that should handle the intent, and
- an optional *Extras* field that carries additional information required to accomplish the requested action.

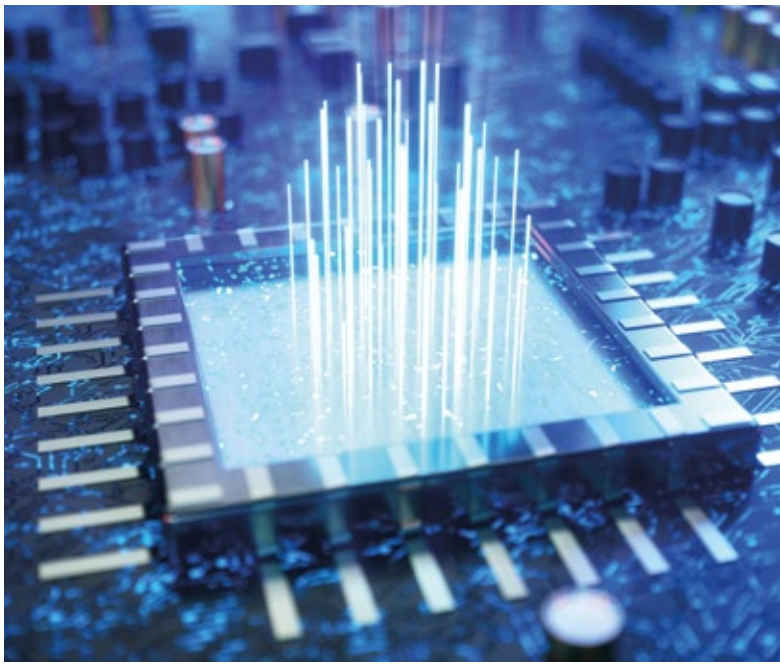
When an Intent is sent from an App to the Android OS, the system uses the first three properties to resolve it. The Android OS checks the available information of the current setup, i.e. manifest files of all currently installed apps, and selects an appropriate app component to start.

B. Information-centric networking

ICN paradigms, of which *Named Data Networking (NDN)* [14] is one of the most popular operational examples, are focused in ways to reach information objects, while in contrast today's Internet is focused on host reachability. The architectural design of ICN paradigms comprise by design the following aspects:

- receiver-driven asynchronous publish-subscribe communication.
- integrated security, not just in terms of data as well as in terms of data and naming binding.
- flexible and reliable data-centric multipath routing.
- flexible naming space.
- built-in mobility support (interface abstraction, Face, and no use of addresses) [21].

ICN is emerging as a new stack which is being explored in multiple areas. However, it is in the field of IoT that ICN is gaining ground, as explained next.



12 https://developer.android.com/guide/components/intents-filters

13 https://developer.android.com/guide/components/intents-common

14 https://named-data.net/

There are several ICN architectures being explored from an end-to-end perspective, being the most popular ones the *Content Centric Networking (CCNx* [15], 2010), by PARC and partners; NDN (2014), by UCLA and partners; *Hybrid ICN (hICN* [16], 2014), by Cisco. CCNx, originally developed by PARC, gave rise to both NDN and hICN. All of the 3 mentioned software architectures are therefore quite similar, being the main difference the fact that hICN is an adaptation of ICN to IP (overlay approach). While CCNx and NDN can run directly on top of the OSI MAC Layer.

The original CCN has been adopted by the IRTF working group ICNRG[17]. Moreover, in what concerns relevancy for IoT, the most popular approach today is NDN as it supports all IoT basic requirements [20].

The ICN/NDN architecture embodies a publish/subscribe pull-based communication model. Producer nodes correspond to devices that send data (*Data* packets), once they get an expression of interest by consumer nodes (*Interest* packets). Data packets are sent back following ICN forwarding strategies, and based on the network state (*breadcrumbs*) left by Interest packets in routers along the way.

Moreover, NDN follows a *store-and-forward principle* and hence, any node in the network acts as an NDN router, and holds three different data structures: the *Forwarding Information Database (FIB)*, the PIT, and the *Content Store (CS)* database.

The FIB holds aggregated name prefixes for data objects matching outgoing *Faces* (interface abstraction). It should be highlighted that a Face can interconnect to different networking technologies, as well as to different applications, services, etc.

To fetch content, a consumer sends an Interest packet to the network containing the name of the required content. When an NDN node receives an Interest message, it first queries matching data in its local CS. If the data is locally available, a matching Data packet is sent back to the consumer through the same Face. Otherwise, the node updates its PIT table with the Interest packet name prefix, associated to the incoming Face.

If there is no match in the PIT, then the node forwards the Interest packet further over the recorded outgoing interface(s) in the FIB. When the Interest packet reaches a potential data provider or a node having a matching Data packet in its CS, a Data packet is generated and replied back to the consumer, following the chain of the intermediate nodes. During the forwarding process, each node replicates the Data packet to all recorded incoming interfaces in the matching PIT entry, keeps a copy in the local CS, and then deletes the related PIT record. Thus, NDN traffic is self-regulated, and in each link, for the same object, there is at most one Data and one Interest packet.

The operation of NDN is therefore based on a pull-model, where consumers first express interest about a specific object. Nevertheless, NDN supports a second model, push-based, derived from applications where data can be directly pushed to multiple consumers, without having these specifically expressing an interest before. As NDN is a network layer solution, push based models can be implemented by applications in a variety of ways [22].

Furthermore, in large-scale scenarios NDN provides Interest aggregation within the PIT structure (aggregation of multiple Interest requests onto a single aggregate request). In-network caching allows consumers to retrieve cached content from intermediate routers, and not necessarily from producers. The different forwarding strategies (e.g., anycast) allow NDN to take into consideration availability and restrictions of devices.

Summarising, today there are several relevant approaches to provide interoperability at the network layer. These approaches are content-centric or named-data based, and take into consideration application requirements, as well as context-awareness, to best adjust the networking infrastructure to the needs of services and of critical environments.

15 https://github.com/ProjectCCNx/ccnx

16 https://fd.io/docs/hicn/latest/

17 https://datatracker.ietf.org/rg/icnrg/documents/

V Active Research Directions in fortiss IIoT

A. Automated Translation of Skills to WoT TD
The WoT TD provides a way to describe interfaces and meta-data of a Thing. It provides three *InteractionAffordances* types that describe how a potential entity can interact with a Thing:

- *PropertyAffordance* for describing the Thing states;
- *ActionAffordance* for invoking the Thing functions to change a state;
- *EventAffordance* for pushing the information from a provider to a consumer, in an asynchronous manner.

A skill representation of functionalities of a PLC, as described in section III, provides an interface that allows other entities to interact with it. The possibility to transform a skill-based engineering model into one or multiple WoT TDs by means of *InteractionAffordances* is relevant to integrate brownfield devices in the overall Edge-Cloud IIoT environments. However, such a transformation is not trivial. For instance, the concept of a state machine includes methods to be invokable only when the machine is in a certain state, i.e., the start method can only be triggered when the machine is in the idle state. The WoT community already started discussing that dynamic behavior of *InteractionAffordances*[18], but does not include aspects regarding legacy device integration.

Therefore, research questions being addressed in this context are:

- How can *Thing Descriptions* be extended in order to express all relevant PLC components? This refers to the capability to model the dynamic behavior of methods in a state machine.
- How can *Thing Descriptions* be generated automatically based on existing elements, i.e. PLC program code, interface definition etc.? Current approaches require the integration of specific domain models, or tools such as AutomationML to assist such a transformation between an abstract PLC representation, i.e. skill-based model, and a WoT TD. What other possibilities can be considered?

18 https://github.com/w3c/wot-thing-description/issues/899

A proposal being followed is to consider a transformation as follows (rf. to section VI for further operational aspects):

- The state machine and its states can be mapped to the WoT TD *PropertyAffordances*.
- The control methods could be transformed into *ActionAffordances*.
- The transition event of the state machine are mapped to the *EventAffordances*.

B. Building Intents in an Automated Way
Most of the existing, proposed solutions towards Intent-based Networking focus on enabling a facilitated management of enterprise networks and therefore support the tasks of network administrators [23]. Even though network management is a relevant field also in IIoT, we want to explore Intents in order to enable an automated adaption of network functions based on *Things'* requirements. Therefore, we consider Things as *Intent Users* that describe their requirements towards the network. As Things are not able to express Intents in natural language, for example, other forms to represent their requirements are needed.

One possibility here is to use meta-information that is associated with the IoT application. For example, Thuluva et.al. introduce *Recipes* that act as description of how *Ingredients*, i.e. capabilities of Things, shall interact with each other in order to compose an IoT application [24]. Derived from such a *Recipe* description, an Intent can be created that specifies the communication requirements between Things, i.e., routing paths. The Intents are sent to the network, i.e., to a mediator such as an SDN controller, or components such as switches and routers directly, that trigger low-level actions, e.g., an adaption of routing tables, in order to ensure the required behavior.

In that sense, the following research questions are being addressed:

- What information is required from Things, i.e. sensors, machines, and services, that would enable the network to support their operation? In other words, what meta-data is relevant to be sent from Things to the network so that it can self-adapt?
- How can the specific requirements be expressed via Things capabilities?

An answer to these first questions would lead to a technical specification of an Intent abstraction for IoT.

Especially in the context of legacy devices, such an approach would need to consider existing technologies and their constraints that are deployed in a legacy system setup as an adaption of such established technologies is hard to implement.

- Which network mechanisms, i.e. protocols, functions, can be used to submit an Intent and thereby inform the network about the required behavior?

If possible, existing mechanisms should be (re-)used here so that an integration of the proposed solution into an existing network environment is more likely.

C. ICN and IoT

An advantage of ICN Publish-Subscribe models is decentralization. From a network architectural perspective, such models are promising candidates for data transmission in highly heterogeneous IoT scenarios. The ICN Publish-Subscribe semantics support integrated security and data distribution/decentralization via in-network caching. Although the transient nature of IoT data may bring challenges related to in-network caching, new research findings corroborate that caching techniques specifically designed for small transient data can actually reduce the time-to-completion of requests [25][26]. Furthermore, the ICN interface abstraction model, Face, is extremely relevant in supporting the sharing of data between devices, as well as between applications and services.

Despite its advantages, by design, the ICN Publish-Subscribe paradigm follows a pull-based communication approach, where consumers need to express interest to receive each data packet. Such a pull-based model may reduce performance in scenarios holding a large number of resource-constrained, mobile devices, as occurs in IoT environments, due to the need to frequently transmit Interest packets. This could require devices to be in reception mode all the time, for instance, thus resulting in battery drain. Or, it could require a method for fine-grained synchronization, based on the rate at which individual IoT devices emit data. Therefore, a few research items that are being addressed in IIoT are:

- **Push models for IoT.** The NDN design is based on a pull model, which may not be enough in the context of IoT environments, for instance, in situations where devices emit periodic data (periodic triggered data transmission), or in alarm situations (event-triggered data transmission). Push based communications are required in such situations and relevant to support a faster data forwarding. There are several possibilities being advocated to implement push-based communication, each of them with a specific tradeoff, but performance for specific IoT scenarios needs to be evaluated [22].

- **The need to support multi-party data synchronization.** The NDN architectural design considers ways to transfer data from N producers to M consumers, in a way that is data-centric and scalable. NDN guarantees delivery even in the verge of intermittent connectivity; it does not, however, guarantee synchronization of data among consumers. Such synchronization is usually required in environments where a group of N nodes keeps a shared data set. Synchronization may also be required in IIoT situations involving critical data. For instance, support for remote management may require strong synchronization among control devices. The original design of CCN references a Synchronization Protocol (SYNC) [27]. The most recent evolution of NDN synchronization protocols is the VectorSync protocol [28] which is based on the Sync principles, but improves data synchronization and considers group membership management.

- **In-network caching strategies.** In-network caching in NDN follows the widely research principles developed for wired networks. However, in the context of IoT scenarios, it is necessary to consider constrained and heterogeneous devices, as well as mobility of devices, aspects which were not supported by design in the original protocols. Data is therefore transient, it expires after a specific time interval, aspect that regular random caching and reference caching approaches did not address.

- **Producer mobility support.** NDN naturally supports consumer mobility, and this type of mobility is the most common case for IoT scenarios. For scenarios involving PIoT, and scenarios involving direct communication between IoT devices/relays (such as autonomous vehicles), there is the need to support producer mobility as well. There are currently two type of solutions being proposed to address producer mobility: i) de-centralized, anchor-less solutions where producers send alerts after moves occur (late-binding); ii) in-network caching policies for producers. Solutions that attempt to optimize in-network caching are the second type of approach where producers cache the content "around". A possible approach to optimize placement, having in mind IoT scenarios such as connected vehicles, is to consider context-awareness and measures for the producer's neighborhood: centrality, availability of neighbors, as well as similarity. A detailed debate on producer mobility aspects can be found in [21].

- **Naming spaces suitable for IIoT.** The hierarchical naming supported by NDN is flexible and allows to treat both data and services in an equal way, agnostic from devices. It is application expressive and as such, NDN does not need to consider a name resolution service such as DNS (even though external name discovery services can be used). Naming hierarchy is, however, not a strict requirement. It is used to assist forwarding as well as security, aggregation, and other critical features. Furthermore, NDN allows the use of flat names, which are simply a special case of hierarchical names that just have one component. Even though NDN adopts 1-dimensional hierarchical naming and lookup operations based on longest-prefix matching, for IoT environments users may want to express requests with multi-dimensional attributes. This is a highly relevant feature in the context of IoT applications as data is transient, i.e., it can be bound to specific location and to specific periods of time. The translation of semantics based, multi-dimensional naming into the longest-prefix match followed by NDN is not trivial and is still under development [29]. Multi-dimensional naming is more relevant in the context of vehicular networks, for which bi-directional approaches (with location) have been considered [30]. Although flexible, the NDN naming is still in an early development phase, where there is not yet a consensus in semantics adoption. Due to this, NDN names that have rich application semantics may be an issue, as they may leak individual information behavior, for instance time and location data. If several packets are collected, then the user identity may be endangered, and this is a relevant area to consider in the context of mobile IoT environments. Obfuscation is feasible, but it cannot endanger other NDN functionality, such as name-based routing.

VI The fortiss IIoT Lab Brownfield Device Integration Demonstrator

The demonstrator described in this section is being developed in the context of the fortiss IIoT Lab[19], to explore the concepts and research questions debated in this white paper.

The IIoT Lab is a playground of fortiss for training, experimentation, and novel concept validation. It is based on a series of self-contained demonstrators covering different vertical markets of IIoT, such as Smart Factories, Smart Cities, Smart Facilities.

19 https://www.fortiss.org/forschung/living-lab/detail/iiot-lab

The demonstrator described in this section is focused on the applicability of IIoT in Smart Factories, and specifically on aspects concerning an automated brownfield device integration in IIoT environments (rf. to the questions in section V.A.). The section starts with an exemplar use-case, and then provides details on the testbed being set, its actors, and goals.

A. Exemplar Use-case

Thomas is a production manager at a factory. Due to globalization and increased market demands, the company that Thomas works for needs to optimize its production. For this, Thomas is investing in an IIoT open-source system, which provides improved efficiency via data analysis on the Cloud. However, part of the equipment in the factory is not supported by the acquired Cloud computing platform. This is the case, for instance, of a sorting machine for which downtime causes and their impact are not clearly understood. The machine sometimes stops operating with no obvious reason. The connected machine terminal then states that the sorting container is full even if it is not. This is because the internal memory of the machine counts the number of produced pieces. When the counter exceeds a certain limit, the machines stops. Usually, a human operator needs to walk over, acknowledge the warning and force the machine to continue. Thomas believes that this type of situation can be circumvented via the integration of the machine into the IIoT acquired platform, meaning first that additional sensors can assist in a more quick detection of downtime, and second, that such platform could reduce the need for manual intervention.

For that purpose, on the IIoT gateway, an element that is part of the acquired IIoT platform, the IT department installs a new solution under developed by fortiss, coined in this description as **BFThing**. BFThing provides a way for a legacy device to be integrated into open-source IIoT systems via an automated PLC description into a WoT.

Via this novel software module, an Edge/Fog device, or an IIoT gateway shall be able to support bi-directional connectivity to brownfield devices. Standardized communication protocols and data models from IIoT domain as well as conversion tools to integrate legacy devices facilitate the connection buildup. Thereby, a smooth and seamless connectivity is established.

B. Technical Description

A high-level illustration of the demonstrator is provided in Figure 3. An Edge node (in our case, an IIoT gateway) hosts BFThing, the novel middleware developed by fortiss. This middleware can be loca-

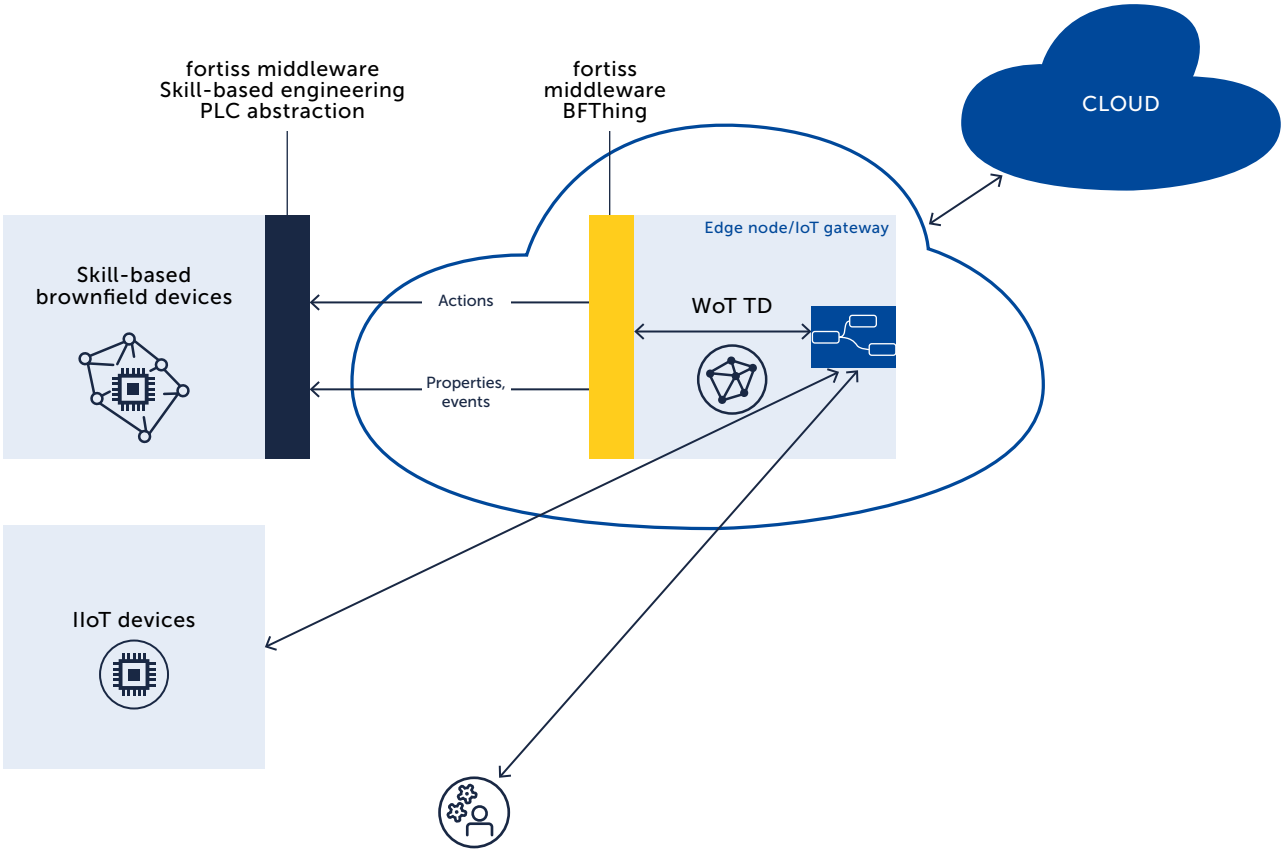


Figure 3: Automated integration of brownfield devices via the fortiss BFThing middleware.

ted on the Edge, or on the Cloud. BFThing receives input (currently via OPC-UA; on a later stage agnostic to the underlying protocol) from the field-level skill-based PLC abstraction model and provides an automated transformation to a WoT TD. The left-hand skill-based engineering middleware converts the proprietary technologies to domain-specific standards, such as OPC-UA. Thereby, it is possible to create a formal description of the machine's data and functionalities using skill-based modeling approaches. The information gathered and transformed by BFThing into a standardized WoT format includes the machine's type, the available data, data types and properties, a description of the invokable actions, and how to call them. BFThing, therefore, handles an automated translation of the skill-based engineering abstraction middleware to a WoT TD format, and also handles the bi-directional communication to brownfield devices. It instantiates all needed communication channels, e.g., data subscriptions and communication clients. As illustrated, in a first step, the output of the description of the field-level device will be combined, derived from BFThing output to e.g. a WoT node/Node-Red, with other measurement values coming from IIoT sensors. For instance, the machine emits a message notifying that the fill container is full. The sensors provide measurement concerning such level. By relying on BFThing, it is feasible to fuse measurements from the PLC and additional sensors thus improving the accuracy in terms of potential measurements. As also illustrated, in this demonstrator the outcome will be sent to a personal device (e.g., a rugged tablet) of a human operator, where an API developed by fortiss will provide the visualization of results and also access to specific actions to be triggered by a human operator. For instance, the person can acknowledge the warning and force the machine to continue, by clicking a specific button which triggers an action via the middleware on the IoT gateway. The information about the available actions and how to trigger them is retrieved from the machine's TD and is sent to a personal device carried by the user.

VII Summary

This white paper provides an overview on current semantic interoperability paths being tackled in the IIoT area, by fortiss. Semantic technologies are being used to automate the integration of OT systems to end-to-end IT systems, via a full and automated interconnection of Things for different vertical domains. For that purpose, there are several aspects currently being addressed based on a semantic object notion derived from the WoT recent standards:

- integration of brownfield devices via an automated transformation of a skill-based engineering description into a WoT description format.
- adaptation of the network services via intent-based networking.
- development of "Intents" in a way that assists in bringing application level objectives into the network layers.
- exploitation of information-centric networking paradigms for a decentralized Publish-Subscribe data exchange, such as the receiver-driven publish-subscriber approach from ICN/NDN.

VIII References

[1] R. M. Zein Nader (editor), "IEEE Nendica Report: Flexible Factory IoT: Use Cases and Communication Requirements for Wired and Wireless Bridged Networks," in IEEE-SA Industry Connections Report. IEEE 802.1-19-0026-06, 2020.

[2] M. McCool, M. Kovatsch, T. Kamiya, V. Charpenay, and S. Kabisch, ¨ "Web of things (WoT) Thing Description," W3C, W3C Recommendation. April 2020. Available at: https://www.w3.org/TR/2020/REC-wot-thingdescription-20200409/.

[3] E. Korkan, H. B. Hassine, V. E. Schlott, S. Kabisch, and S. Steinhorst, ¨ "Wotify: A platform to bring web of things to your devices," arXiv preprint arXiv:1909.03296, 2019.

[4] R. Mosshammer, A. Einfalt, A. Lugmaier, J. Hodges, and F. Michahelles, "Semantic annotation engine for smart grid applications," inProc. 5th International Conference on the Internet of Things (IOT). IEEE, 2015, pp. 132–137.

[5] L. Racchetti, C. Fantuzzi, L. Tacconi, and M. Bonfe, "Towards an ` abstraction layer for plc programming using object-oriented features of iec61131-3 applied to motion control," inProc. IECON 2015 - 41st Annual Conference of the IEEE Industrial Electronics Society, pp. 000298– 000303, 2015.

[6] N. E. Petroulakis, E. Lakka, E. Sakic, V. Kulkarni, K. Fysarakis, I. Somarakis, J. Serra, L. Sanabria-Russo, D. Pau, M. Falchetto et al., "Semiotics architectural framework: End-to-end security, connectivity and interoperability for industrial iot," inProc. in IEEE Global IoT Summit (GIoTS)., pp. 1–6. 2019.

[7] E. Korkan, S. Kaebisch, M. Kovatsch, and S. Steinhorst, "Safe interoperability for web of things devices and systems," inProc. in Languages, Design Methods, and Tools for Electronic System Design. Springer, 2020, pp. 47–69.

[8] M. Behringer, M. Pritikin, S. Bjarnason, A. Clemm, B. Carpenter, S. Jiang, and L. Ciavaglia, "Autonomic Networking: Definitions and Design Goals," Internet Requests for Comments, RFC Editor, RFC 7575, June 2015. [Online]. Available at: http://www.rfc-editor.org/rfc/rfc7575.txt.

[9] A. Clemm, L. Ciavaglia, L. Granville, and J. Tantsura, "Intent-Based Networking - Concepts and Definitions,". Internet draft, Network Working Group, Expired. Available at: https://tools.ietf.org/html/draft-irtf-nmrg-ibn-concepts-definitions-01.

[10] C. Li, O. Havel, W. Liu, P. Martinez-Julia, J. Nobre, D. Lopez. "Intent classification". IETF draft, Network Working Group, Informational. Nov 2019 (expired). Available at: https://tools.ietf.org/html/draft-li-nmrg-intent-classification-02.

[11] H. van der Veer and A. Wiles, "Achieving Technical Interoperability the ETSI Approach," European Telecommunications Standards Institute, Tech. Rep., 04 2008.

[12] M. Bauer, H. Baqa, S. Bilbao, A. Corchero, L. Daniele, I. EsnaolaGonzalez, I. Fernandez, O. Franberg, R. Garcia Castro, M. Girod-Genet, ¨ P. Guillemin, A. Gyrard, C. El Kaed, A. Kung, J. Lee, M. Lefranc ¸ois, W. Li, D. Raggett, and M. Wetterwald, "Semantic iot solutions - a developer perspective," Tech. Rep., 10 2019.

[13] E. Lakka, N. E. Petroulakis, G. Hatzivasilis, O. Soultatos, M. Michalodimitrakis, U. Rak, K. Waledzik, D. Anicic, and V. Kulkarni, "End-to-end semantic interoperability mechanisms for iot," in 2019 IEEE 24th International Workshop on Computer Aided

Modeling and Design of Communication Links and Networks (CAMAD), 2019, pp. 1–6.

[14] K. Dorofeev and A. Zoitl, "Skill-based Engineering Approach using OPC UA Programs," in IEEE 16th International Conference of Industrial Informatics (INDIN), Jul. 2018.

[15] P. Zimmermann, E. Axmann, B. Brandenbourger, K. Dorofeev, A. Mankowski, and P. Zanini, "Skill-based Engineering and Control on Field-Device-Level with OPC UA," in Proceedings of the IEEE International Conference on Emerging Technologies And Factory Automation (ETFA), Sep. 2019.

[16] A. Bröring,J. Seeger, M. Papoutsakis, K. Fysarakis, and A. Caracalli, "Networking-aware IoT application development," MDPI Sensors, vol. 20, no. 3, p. 897, feb 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/3/897.

[17] Y. Elkhatib, G. Coulson, and G. Tyson, "Charting an intent driven network," in 2017 13th International Conference on Network and Service Management (CNSM), 2017, pp. 1–5.

[18] A. Elhabbash, G. S. Blair, G. Tyson, and Y. El-khatib, "Network conscious edge service adaptation," 2018.

[19] Q. Technologies, "LTE Direct Always-on Device-to- Device Proximal Discovery," no. August, pp. 1–13, 2014.

[20] M. Meddeb, A. Dhraief, A. Belghith, T. Monteil, K. Drira, and S. AlAhmadi, "Named data networking: A promising architecture for the internet of things (iot)," International Journal on Semantic Web and Information Systems (IJSWIS), vol. 14, no. 2, pp. 86–112, 2018.

[21] Rute C. Sofia, Guidelines towards Information-Driven Mobility Management. MDPI Future Internet, 11(15)111, DOI: 10.3390/fi11050111. May 2019.

[22] R. C Sofia and P. M Mendes, "An overview on push-based communication models for information-centric networking,". MDPI Future Internet, vol. 11, no. 3, p. 74, 2019.

[23] L. Pang, C. Yang, D. Chen, Y. Song, and M. Guizani, "A Survey on Intent-Driven Networks," IEEE Access, vol. 8, pp. 22862–22873, 2020. [Online]. Available: https://ieeexplore.ieee.org/document/8968429.

[24] A. S. Thuluva, A. Broring, G. P. Medagoda, H. Don, D. Anicic, ¨ and J. Seeger, "Recipes for IoT applications," in Proceedings of the Seventh International Conference on the Internet of Things - IoT '17. New York, New York, USA: ACM Press, oct 2017, pp. 1–8. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3131542.3131553.

[25] S. Vural, P. Navaratnam, N. Wang, C. Wang, L. Dong, and R. Tafazolli, "In-network caching of internet-of-things data," in 2014 IEEE International Conference on Communications (ICC). IEEE, 2014, pp. 3185– 3190.

[26] S. Vural, N. Wang, P. Navaratnam, and R. Tafazolli, "Caching transient data in internet content routers," IEEE/ACM Transactions on Networking, vol. 25, no. 2, pp. 1048–1061, 2016.

[27] H. B. Abraham and P. Crowley, "Performance measurement of the ccnx synchronization protocol," in Architectures for Networking and Communications Systems. IEEE, 2013, pp. 121–122.

[28] W. Shang, A. Afanasyev, and L. Zhang, "Vectorsync: distributed dataset synchronization over named data networking," in Proceedings of the 4th ACM Conference on Information-Centric Networking, 2017, pp. 192– 193.

[29] S. Gao, H. Zhang, and B. Zhang, "Supporting multi-dimensional naming for ndn applications," in 2016 IEEE Globecom Workshops (GC Wkshps). IEEE, 2016, pp. 1–6.

[30] D. Saxena, V. Raychoudhury, N. Suri, C. Becker, and J. Cao, "Named data networking: a survey," Computer Science Review, vol. 19, pp. 15– 55, 2016.

Imprint

Publisher
fortiss
www.fortiss.org

Authors
Kirill Dorofeev
Hendrik Walzel
Prof. Dr. Rute C. Sofia

Layout
Sonja Taut

Print
viaprinto | CEWE Stiftung & Co. KGaA
Martin-Luther-King-Weg 30a
48155 Münster

ISSN Print
2699-1217

ISSN Online
2700-2977

1. Edition, January 2021

Picture credits

Titel: shutterstock ©ZinetroN
Seite 4: shutterstock © PopTika
Seite 6: shutterstock © Blackboard
Seite 8: shutterstock © DrHitch
Seite 13: shutterstock © Ustyna Shevchuk
Seite 16: shutterstock © ESB Professional
Seite 18: shutterstock © spainter_vfx
Seite 22: fortissGmbH ©Kathrin Kahle

Acknowledgements
This work has been partially supported by funding from the BMWi project *Mittelstand 4.0-Kompetenzzentrum Augsburg*, reference nr 01MF16002E, 2020.

fortiss is the Free State of Bavaria research institute for software-intense systems based in Munich. The institute's scientists work on research, development and transfer projects together with universities and technology companies in Bavaria and other parts of Germany, as well across Europe. The research activities focus on state-of-the-art methods, techniques and tools used in software development and systems & service engineering and their application with cognitive cyber-physical systems such as the Internet of Things (IoT).

fortiss is legally structured as a non-profit limited liability company (GmbH). The shareholders are the Free State of Bavaria (majority shareholder) and the Fraunhofer-Gesellschaft zur Förderung der angewandten Forschung e.V.

fortiss GmbH
Guerickestraße 25
80805 Munich
Germany
www.fortiss.org
Tel.: +49 89 3603522 0
E-Mail: info@fortiss.org

fortiss