Received June 20, 2021, accepted July 5, 2021, date of publication July 9, 2021, date of current version July 16, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3096043

Model-Driven Interoperability Layer for Normalized Connectivity Across Smart Grid Domains

ALAA S. ALAERJAN
Department of Computer Science, College of Computer and Information Sciences, Jouf University, Sakaka 72341, Saudi Arabia
e-mail: asalaerjan@ju.edu.sa

This work was supported by the Deanship of Scientific Research at Jouf University under Research Grant DSR2020-02-2608.

ABSTRACT The connectivity of heterogeneous components is the key factor behind the notion of the Internet of Things (IoT). Typically, IoT applications involve several communication protocols that are developed based on heterogeneous data models. This has complicated the connectivity within IoT applications. It has also caused significant interoperability issues. Therefore, in this paper we propose a novel connectivity layer which we refer to as the Distributed Data Interoperability Layer (DDIL). DDIL aims at addressing the connectivity issues that arise due to the heterogeneity of data models. In the approach, we construct DDIL into different software components. We then describe these components as a set of configurable features to allow DDIL to be tailored based on the requirements of each application. DDIL has the capabilities to address both syntactic and semantic interoperability. The feature-oriented design of DDIL provides required flexibility which is a key concern in several IoT applications. Additionally, DDIL supports backward compatibility. It also allows utilizing preexisting technologies which supports rapid development of applications. We implemented the approach in a simulated smart grid environment. The results prove that DDIL has the capabilities to support the connectivity of different applications even if they are developed based on different protocols and heterogeneous data models.

INDEX TERMS Applications, connectivity, feature-oriented, interoperability, IoT, model-based.

I. INTRODUCTION

The Internet of Things (IoT) has emerged due to the rapid development in information and communication technology (ICT). Several IoT applications (e.g., smart grids, smart cities) start to realize the endless opportunities for improving quality concerns such as productivity, reliability, and efficiency. IoT promises to overcome the challenges that are currently being faced by several industrial and civil applications. IoT aims at addressing the lack of monitoring and control. It also aims at improving operational and predictive capabilities [1]. IoT depends on connectivity to allow applications to integrate physical objects with ICT [2]. This integration enables communicating components to exchange real-time information which improves domain awareness and enhances decision making [3].

A cornerstone of the IoT notion is connecting different domains within IoT applications. This requires extensive efforts on understanding the connectivity nature of IoT applications. Several IoT applications are considered as modernization of current applications. For instance, smart grids are modernization of current power grids [4]. Therefore, any connectivity approach must consider the nature of current applications as well as the requirements of modernized IoT applications. Given that, one of the major challenges that are currently being faced by industrial applications is communication interoperability [5]–[7]. This challenge also affects several IoT applications. The communication interoperability issues deteriorate the connectivity in IoT applications and force different domains to operate in isolation which contradicts the vision of IoT. In fact, interoperability is a prerequisite for connectivity in large-scale systems, and connectivity is not achievable without addressing communication interoperability issues [8], [9].

There have been several proposals (e.g., The Industrial Internet of Things Connectivity Framework, Volume G5 [10]) for standardizing interoperability in IoT applications. However, many of those proposals abstractly describe the solutions which does not totally resolve the current connectivity issues. Typically, different domains within IoT applications

The associate editor coordinating the review of this manuscript and approving it for publication was Bin Zhou.

use different communication protocols. Each protocol uses its own data model which is inherently different in terms of structure and design. Additionally, the implementation of data models is subjective to vendors interpretation [11]. As a result, the interoperability of domains is significantly affected. Hence, the overall connectivity of IoT applications is deteriorated.

In current practices, several applications are built upon protocols that supports machine-to-machine communication such as OPC-UA. These protocols are designed to support local connectivity [12] which means they can not support systems scalability. On the other hand, protocols such as the Data Distribution Service (DDS) [13] are designed to enable scalability and perform autonomously [14]. Unfortunately, the data models of OPC-UA and DDS are heterogeneous. Therefore, it is not feasible to allow applications that are developed based on these protocols to exchange information. This type of interoperability greatly affects connectivity in future IoT applications. According to a report by Manyika *et al.* [15], interoperability of communicating components accounts to about 40%-60% of the total value of IoT applications. It is worth mentioning that in this paper the term ''IoT application'' indicates a system that integrates physical objects, software, and communication technologies (e.g., smart grids, smart cities, smart homes). On the other hand, the term ''application'' means the software that is used to perform specific functionalities (e.g., Supervisory Control and Data Acquisition [SCADA], Energy Management System [EMS]).

In this paper, we present a novel layer for supporting the connectivity in IoT applications. We refer to it as the Distributed Data Interoperability Layer (DDIL). DDIL aims at addressing the communication interoperability issues that occur due to the heterogeneity of data models. DDIL also aims at maximizing connectivity by allowing different applications to exchange data even if they use different communication protocols. In the approach, we combine two software development techniques which are Model-Driven Engineering (MDE) and Feature-Oriented Modeling (FOM). We use MDE to develop the components of DDIL into a set of models. We then use FOM to allow DDIL models to be configurable based on the requirements of each application. MDE and FOM are used together because IoT applications consists of a wide variety of applications and devices which requires flexible connectivity solution to support rapid development and integration [1], [3].

DDIL is developed to be placed under the application layer and on top of communication protocols to maximize connectivity. We experimented the implementation of DDIL on a simulated smart grid environment. The results show that DDIL supports connectivity even when it is used with heterogeneous ICT. Based on the above discussion, we observed that introducing an interoperability layer on top of common networking layers greatly benefits different domains in IoT applications. Therefore, we summarize our contributions as follows:

1) Providing DDIL which is an abstraction layer that combines two different development techniques (i.e., MDE, FOM) to support interoperability across different IoT domains. DDIL permits configurability which allows different applications to utilize the same layer yet tailor it based on the required features. In this work, we systematically define the tailoring points by providing a configurable feature model.
2) Defining several software models based on the notion of separation of concerns to address connectivity issues among IoT applications. Particularly, six models are defined to address data transformation, data semantics, data naming, and data integration issues. Those models allow applications to exchange data even if they are built upon heterogeneous protocols.
3) An implementation of the proposed layer is presented to demonstrate the ability of DDIL to connect applications that are built upon heterogenous protocols in a smart grid. The demonstrated use case allows four different applications that are built upon three heterogenous protocols (i.e., DDS, MQTT, OPC-UA) to seamlessly exchange data.

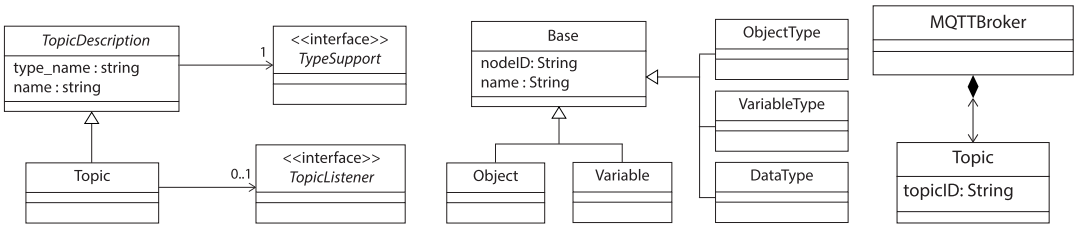
The rest of this paper is organized as follows. Section II describes the connectivity challenges in IoT application. It also provides an IoT context example to be used throughout this paper. Section III provides an overview of some related work. Section IV describes DDIL structure, models, and design principles. Section V provides a use case illustrating the viability of DDIL. Section VI provides an evaluation of DDIL and describes the challenges that need to be addressed in order to adopt DDIL. Finally, Section VII concludes this paper and provides a discussion on future work.

II. IoT CONNECTIVITY ISSUES

This section provides an overview of the connectivity issues that hinder the full realization of IoT applications and their potentials. It also describes a context example to illustrate the challenges in a smart grid which is one of the major IoT application.

A. CHALLENGES

The rapid development of new technologies has enabled IoT applications such as smart grids and smart cities to improve efficiency of work, safety, and security. However, there is still a demand for developing flexible connectivity solutions for IoT applications to enable seamless data exchange [5]. IoT applications consist of multiple domains. For example, a smart grid consists of domains such as power generation, distribution, and consumption. A major requirement for smart grids is enabling connectivity among objects (e.g., devices, applications, sensors) not only within the same domain, but also with other domains [3]. This is still a challenging task due to the heterogeneity of communication protocols and data models [1]. In fact, seamless connectivity is required in all IoT applications not only smart grids, and the endless



(a) DDS Data Model (b) OPC-UA Data Model (c) MQTT Data Model

FIGURE 1. Different data models of common IoT communication protocols.

opportunities of IoT applications can not be realized without addressing the connectivity issues.

Interoperability is a key challenge in realizing the connectivity of IoT applications [7]. Typically, an IoT application consists of several communication protocols each of which uses a different data model [16]. This has caused significant interoperability issues and consequently deteriorates the connectivity of IoT applications. For instance, consider data generated from a protection sensor in a smart grid. The same data is represented differently in a different domain because different communication protocols are used. Even when the same protocol is used, the representation of the data might be inconsistent between different implementation since structuring the data is subjective based on the interpretation of each vendor [11], [17]. Based on that, applications in IoT domains are forced to operate in isolation or use an ad-hoc solution which is inefficient.

Figure 1 shows the data representations of three common protocols (i.e., DDS, OPC-UA, MQTT) in IoT applications [6], [18]. The figure illustrates the root cause which attributes to several interoperability issues. It shows that the data models of these protocols are inherently different. For instance, DDS represents data as a structured topic with set of interfaces, types, and properties. OPC-UA represents data as set of hierarchical objects where objects inherent the properties of their superior objects. MQTT represents data as a block of attributes referred to as a topic. Given these differences, assume an application at a control center in a smart grid is connected with three different domains each of which uses a different data model as shown by the figure. Furthermore, assume that each domain sends the same data (e.g., sensor reading) to the control center application. It is evident that there are data interoperability issues at the control application since the data is structured and represented differently among these three data models.

B. IoT CONTEXT APPLICATION

Given the above discussion on IoT connectivity challenges, the following describes the challenges in the smart grid (SG). SG is the future of the electric power grid and it is one of the major applications of IoT [19]. It aims at leveraging ICT to address the challenges that are currently being faced by the current power system. SG depends on connectivity among different domains to make smarter decisions and to create an advanced energy system [20]. However, the connectivity of different domains is still a major challenge in SG. This is because SG consists of multiple power domains (i.e., operation, generation, distribution, and customer) each of which uses its own connectivity protocols [21]. Additionally, the diversity of data models and standards creates further data interoperability challenges in SG [22]. All these challenges result in deteriorating the connectivity of different power domains in SG. Throughout the rest of this paper, we use SG as an example for an IoT application.

Figure 2 illustrates the connectivity issues among different domains in SG. It shows two types of interoperability challenges: vertical and horizontal. The vertical interoperability issues occur between lower domains (i.e., generation, distribution, customer) and management domain (i.e., operation). On the other hand, the horizontal interoperability issues occur within the operation domain. All these issues are due to the fact that each domain in SG uses its own protocols and data models. For example, the customer domain depends on lightweight communication protocols such as ZigBee and MQTT [23]. These protocols produce data based on their structured models. When power operators try to route data from customer domain to cloud or operation domains, several interoperability issues arise due to protocols heterogeneity [24], [25]. The same issues occur between operation applications and generation/distribution applications. Regarding the horizontal interoperability issues, they occur between control systems at the operation domain since those systems involve modern automation protocols (e.g., DDS) and legacy systems based on protocols such as OPC-UA [26], [27].

Table 1 shows a comparison between common utilized protocols in SG. The protocols are described in terms of Paradigm, Data Model, and QoS provision. The paradigm criterion describes the data exchange mechanism among the communicating components. The data model criterion specifies whether if the data model of the protocol is standardized. The QoS (quality of service) provision criterion indicates whether the protocol is designed to consider quality aspects in communication. From the table, it is obvious that SG protocols are inherently different which illustrates the need for a connectivity solution.

III. RELATED WORK

There has been a great effort towards improving the connectivity of IoT applications. Several works have been trying to

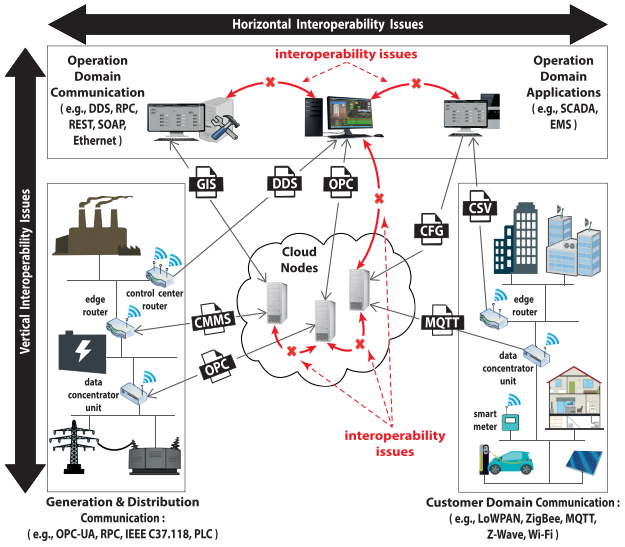


FIGURE 2. Connectivity issues among SG domains.

address the issues of data management among applications. Other works are proposed to enable the connectivity and allow applications to exchange data. The following provides an overview of some related work.

The authors Girau *et al.* [28] propose a cloud based connectivity platform which they refer to as Lysis. The platform enables the development of distributed IoT applications. It is described into four levels architectural model; include real world level, visualization level, aggregational level, and application level. Lysis describe the communicating objects as autonomous social agents. Each agent controls its data and has the ability to autonomously connect with other objects through the virtualization layer. The authors claim that Lysis supports software reusability since it fully exploits the platform as a service (PaaS) model. The work provides a use case illustrating the implementation of Lysis.

The work by Patti *et al.* [29] introduces a distributed infrastructure to support general purpose connectivity in SG. This core purpose of this work is providing a peer-to-peer software infrastructure to enable connectivity across different actors in SG. The infrastructure is a middleware that consists of three different layers which are application, service, and integration. Each layer is responsible for different functionalities. For example, the service layer is responsible for providing the middleware functionalities as web services. The integration layer integrates the data from different actors and provides it to the service layer. The application layer allows applications to developed and integrated with the lower layers. The middleware relies heavily on the publish-subscribe paradigm to connect different actors which imposes restrictions on the communicating components.

The authors in [30] propose an IoT interoperability framework to support communication within the smart home domain. The framework is proposed to address interoperability issues that arise among home area protocols and communication technologies such as ZigBee, Bluetooth, and

TABLE 1. Comparison between utilized protocols in SG.

Protocol	Paradigm	Data Model	QoS Provision
DDS	Publish-Subscribe	Standardized	Yes
OPC-UA	Request-Response	Standardized	No
MQTT	Publish-Subscribe	Standardized	Yes
DNP3	Request-Response	Not Standardized	No
Modbus	Master-Slave	Not Standardized	No
RPC	Request-Response	Not Standardized	No

ZWave. The proposed framework includes three software components. The first is a discovery model which is used for biding and commissioning. The second is a device information model which is used to identify each device within the proposed framework. The third component is a translation model which consists of some submodules to perform the actual translation between the different devices and data representations. The authors demonstrate their approach by showing the sequence of messages within a smart home environment. It is obvious from the work that the approach is designed only to address interoperability issues within a smart home domain and it cannot be generalized to address interoperability issues at larger scale.

The work by Mazayv *et al.* [31] proposes a vision for addressing interoperability issues at the knowledge level. The authors present a semantic profiling framework for discovering the embedded functionalities within an object. These objects then semantically categorized by their interaction patterns. The framework consists of two layers, an upper layer that is used to perform the semantic profiling, and a lower layer used for identifying object objects functionalities. In this work, the authors propose the Constrained RESTful Environments (CoRE) related standards as the key driver for integrating the data. The proposed work is limited in the sense that it targets CoRE related technologies. Additionally, the work requires applications to be smart in nature (i.e., they have embedded smart functionalities) and this is not the case in several IoT domains.

Peña and Fernández [32] propose an architectural model for enabling connectivity among IoT components at three levels fog, edge, and cloud. The core concept of the authors' work is exploiting new technologies (e.g., fog and edge) to enable greater IoT connectivity. The authors summarize their contributions as first, defining the transparency that allows the communicating edge and cloud nodes to dynamically adopt with the changes in the communication topologies. Second, defining the management aspects that govern the connectivity topology to provide a global view for an IoT system. Finally, defining the visualization systems to provide a real-time information about data flow in an IoT system. The work does not fully explain an implementation to validate the proposed architecture. A comparison between the above related works and DDIL is provided in Section VI.

IV. DISTRIBUTED DATA INTEROPERABILITY LAYER

This section describes the Distributed Data Interoperability Layer (DDIL). DDIL is proposed to support

seamless connectivity in IoT applications. It addresses the interoperability issues that occur among heterogeneous data models. Hence, it allows applications to communicate even if they have different communication protocols.

DDIL is a data management and interoperability layer developed based on MDE and FOM. It consists of multiple software components referred to as models. DDIL is implemented under the application layer and above communication protocols (e.g., DDS, OPC-UA, MQTT) as shown in Figure 3. This indicates that DDIL is developed to support multiple data models. DDIL defines all data as objects which means that the basic unit of information in DDIL is a ''data object''. The remaining of this section describes the design decisions and the models of DDIL.

A. DDIL OVERALL DESIGN

The design of DDIL promotes two quality attributes, which are modularity and extendibility. DDIL has been developed with a focus on modularity due to the fact that IoT communication systems are heterogeneous, and software modularity is a key factor for addressing the heterogeneity of communication systems [33]. On the other hand, DDIL considers extendibility due to the fact that most IoT applications are still in a developing stage. Therefore, any proposed solution should consider extendibility to accommodate new systems' requirements [34]. The following describes these attributes in the context of DDIL:

(a) *Modularity:* The structure of DDIL is described into several models as API, Data Source, Data Semantics, Data Naming, Data Integration, Service Configuration, and Quality of Service. These models support the separation of concerns which allows them to handle different aspects of connectivity. In addition, the modular design of DDIL allows for loose coupling among its models which provides flexibility to tailor DDIL based on the requirements of each application. Given that, all DDIL models are built to handle the ''data'' part of the communication. This means that DDIL is built based on the assumption that the underlaying communication structure is already configured to support multiple data models. For instance, if an application is required to participate in two communication protocols (e.g., DDS and OPC-UA), then the assumption is that the underlying communication structure is configured to support both DDS and OPC-UA.

(b) *Extendibility:* DDIL is built to support the extendable nature of applications in IoT domains. Meaning that additional software components can be added without affecting the entire layer. This is mainly because DDIL models are loosely coupled, and they are integrated through a set of APIs. In fact, the extendibility of DDIL supports other qualities such as adaptability which makes it capable of serving across heterogeneous applications.

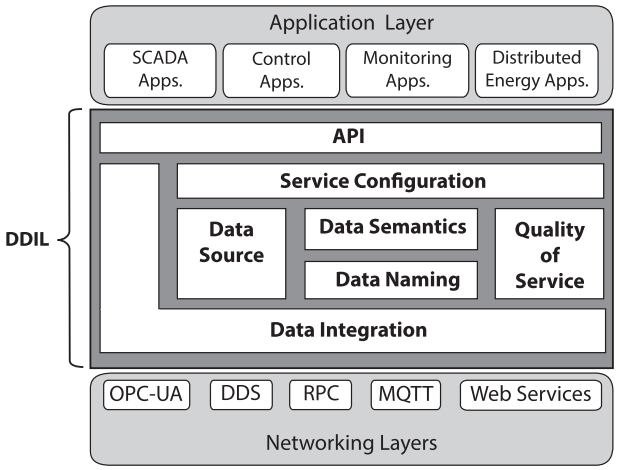


FIGURE 3. Structure of DDIL.

B. DDIL MODELS

DDIL models are described into two categories. Models used to provide functional requirements (i.e., API, Data Source, Data Semantics, Data Naming, Data Integration, Service Configuration), and a model for controlling non-functional requirements (i.e., Quality of Service). The following describes each model in terms of functionalities and relationships.

1) API

The API model is used to integrate applications with DDIL. It provides a set of interfaces to allow an application developer to utilize DDIL without having to know the implementation details of each model. It provides functionalities such as data management, QoS control, and service configurations. The API model eases integrating DDIL with applications and allows developers to focus on developing the applications rather than focusing on the integration process. Eventually, this increases the productivity and reduces applications development cost.

2) SERVICE CONFIGURATION

This model is used by an application to configure DDIL. It is based on FOM [35]. The model provides DDIL services to an application a set of configurable features. A DDIL feature is a software unit that is used to provide either a functional or non-functional requirement. A feature can be grouped with other features to provide a specific configuration. Figure 4 shows the feature model of DDIL. It shows that the root feature is refined into three features as: API, Service_Configuration, and Data_Integration. These features are required which is specified by the filled circles. They are also used together which is specified by the filled triangle under the DDIL feature. The dashed line between the features specifies dependencies. For instance, the use of the Service_Configuration feature depends on the API feature. Thus, in order for an application to use Service_Configuration, it first needs to properly use the API feature.

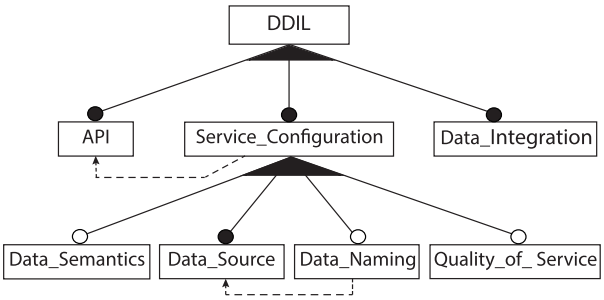


FIGURE 4. Features of service configuration model.

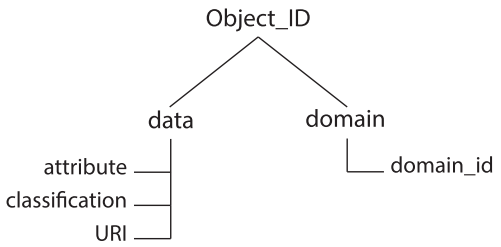


FIGURE 5. Information in an *Object_ID*.

The Service_Configuration feature is further refined into four features as Data_Source, Data_Semantics, Data_Naming and Quality_of_Service. Only the Data_Source feature is required, while the other three are optional which is specified by open circles. The features under the Service_Configuration feature can be used together which is indicated by the filled triangle. The feature model of DDIL is designed to be extendable. If further features (services) are added to DDIL, they can be added to the feature model. For example, if an application requires extra QoS features to be added, then the new features can be defined under the Quality_of_Service feature.

3) DATA INTEGRATION

This model facilitates receiving data with different modeling structures. It allows an application not to be restricted to a specific data model. It provides an integration proxy for each data model. For instance, if an application requires data to be received based on two data models (e.g., DDS topics and OPC-UA objects), then the Data Integration model should provide two proxies one for receiving DDS topics and another for receiving OPC-UA data objects. The Data Integration model acts as a bridge (interface) between the underlying communication protocols and the Data Source model.

The Data Integration model consists of a set of software components that are directly linked with the data models of the underlying communication protocols. The model is built to support transparency, hence when the integration proxy is configured to receive data objects from a specific data model, the application is not aware of how data is received or reconstructed. This takes the burden off developers so they can focus on developing applications rather than developing integration proxies.

4) DATA SEMANTICS

This model is used to define any data semantics required by an application. It provides data utility libraries including data dictionaries. It is coupled with the Data Source model to facilitate data construction and data reconstruction. Based on the defined semantics, the model acts as a validation unit. It ensures that the data produced by the Data Source model follows the semantics defined by the application. For example, suppose that data from an OPC-UA application needs to be propagated to DDS-enabled application. In this case, the transformation from OPC-UA data model to DDS topic is carried out by the Data Source model. However, the validation is performed by the Data Semantics model. The reason for separating out the validation from the Data Source model is because that multiple applications may share the same data model, yet they may require additional semantics or constraints [14]. For instance, in a control center in a power grid, two similar SCADA applications may treat the same data objects differently [36], [37]. The first uses the data to perform a rapid protection response, while the second uses the data for analysis, logging, and storage. Therefore, each application requires extra semantics for the same data.

5) DATA NAMING

This model is used to provide means for uniquely identifying each data object in DDIL. It is required since data object is the basic unit of information. Additionally, because DDIL is capable of supporting heterogeneous data models where different object identification techniques (e.g., Uniform Resource Identifier [URI], key attributes) are used. The model is coupled with the Data Source model to provide the proper object naming. The reason for separating out object naming from the Data Source model is to provide flexibility to an application to internally use its naming technique without affecting the source model of a data object.

DDIL structurally identifies data objects within the notion of a ''data domain''. A domain is used to categorize the type of information that is exchanged among a group of applications. Each data object in a domain has a unique ''Object_ID'' which consists of two parts: one for identifying the data object and another for identifying the domain as shown in Figure 5. Data objects are identified by one of the following techniques: attribute, classification, or URI. An attribute can be used to identify a data object and if used, it must contain a unique value for each data object. The classification can be also used for providing description of a data object. It can be a name or a type that explicitly defines an object. Furthermore, URI can be used to identify a resource that contains a data object. On the other hand, domains are identified by a domain_id. Domain identification (e.g., name, type, number) is created and shared among a group of applications. The setting of a domain_id is performed during the initial configurations of the applications.

6) DATA SOURCE

This is the core model in DDIL since it is responsible for managing data objects. It provides two functionalities which are ''constructing'' and ''reconstructing'' data objects. Constructing data objects is performed based only on the utilized data model of the underlying communication protocol. For example, if an application is configured to participate in DDS communication, then a data object must be constructed based only on the standardized procedure of creating DDS topics. Similarly, if an application is participating in OPC-UA communication, then a data object must follow the standard object creation of OPC-UA. Having said that, data construction is performed only when an application participates in one communication protocol.

Reconstructing data objects on the other hand is performed when an application participates in more than one communication protocol. A data object is reconstructed from a source model to a target model. Once the reconstruction process is completed, the syntax of the reconstructed data object is in compliance with the target data model. Additionally, the reconstructed object is semantically equivalent to the source data object. To illustrate, assume that an application is required to participate in two communication protocols, DDS and OPC-UA. Furthermore, assume that the application is required to pull data from an OPC-UA server and publish the data through a DDS publisher. In this case, OPC-UA data (i.e., source object) is reconstructed into a DDS topic (i.e., target object), and both are semantically equivalent.

In this work, MDE is used for reconstructing data objects. The process follows the three-tiers model transformation approach as shown in Figure 6. A metamodel (M1) represents structured data that is used to describe a specific model (e.g., topic, object). A metametamodel (M2) is represented by a metalanguage and it is used to provide the abstract syntax for the metamodel (M1). A model (M0) represents an instance of the defined structure at level (M1). In this work, the reconstruction of data objects is guided by the following transformation principles:

i *Model-Driven Object Reconstruction:* In order to reconstruct a data object, a standardized approach should be followed to insure consistency. Therefore, both the source model and the target model are defined using Unified Modeling Language (UML) [38].

ii *Semantic Identification of Source and Target Objects:* Prior to performing a transformation, data types of a target model are semantically mapped to source model. This means that the target model describes the semantics of a data object as represented by the source model even when they have different syntax.

iii *Reconstruction Grammar:* Once the syntax and the semantics of source and target data models are identified, transformation grammar is defined and constructed.

iv *Systematic Object Reconstruction:* The reconstruction of data objects from source models to target models should be performed using automatized tools to ensure the consistency of the transformation process which illustrates a need for developing support tools.

Algorithm 1 Data Source Model Operations

```
1:  procedure DSM(InD:in: Input Data, InMdl:in: Source
    Type, Trgt:in: Target Type)
2:      while True do
3:      if InMdl == Trgt then
4:          Construct ← Retrieve Input(InD)
5:          Target_Struct ← Get Dictionary(InMdl)
6:          Object_ID ← Object Naming (Target_Structure)
7:          Execute (Construct, Target_Struct, Object_ID)
8:      else
9:          Reconstruct ← Retrieve Input(InD)
10:         Target_Struct ← Get Dictionary(Trgt)
11:         Object_ID ← Object Naming (Target_Structure)
12:         Execute (Reconstruct, Target_Struct, Object_ID)
```

Algorithm 1 abstractly describes the operations of the Data Source model. The algorithm illustrates that the execution of the procedure Data Source Model (DSM) requires three values, which are input data, source model type, and target model type. If the source type of the input data model is the same as the target type, DSM executes data construction (Construct). DSM first specifies the target structure of the target type (Target_Struct) which should be the same as the source structure. Then, DSM identifies the constructed data object by an (Object_ID) based on the policies defined in the Data Naming model. On the other hand, DSM executes data reconstruction (Reconstruct) if the source type is different from the target type. The same operations that are used for constructing the data are carried for the reconstruction. However, with data reconstruction the target structure is different than the source structure.

7) QUALITY OF SERVICE

This model is used to control data-related QoS requirements. It is used only if the underlying communication protocol is QoS-enabled (e.g., DDS, MQTT). The model allows an application to control QoS requirements at DDIL level. For example, if the underlying communication protocol is DDS, the reliability policy is controlled (tuned) at DDIL level rather than DDS level. This is because data reliability is a concern that is mostly controlled at the application layer more than lower layers. This work provides definitions for three QoS policies since they are supported by some IoT communication protocols:

- *Data Availability:* This policy specifies the state of the data after it has been used in communication. It specifies if a data object should be discarded immediately after it has been sent. It is required when applications produce data at high frequency. This policy is supported by protocols such as DDS.
- *Data History:* This policy specifies how to store data objects after they have been sent/received – keep them

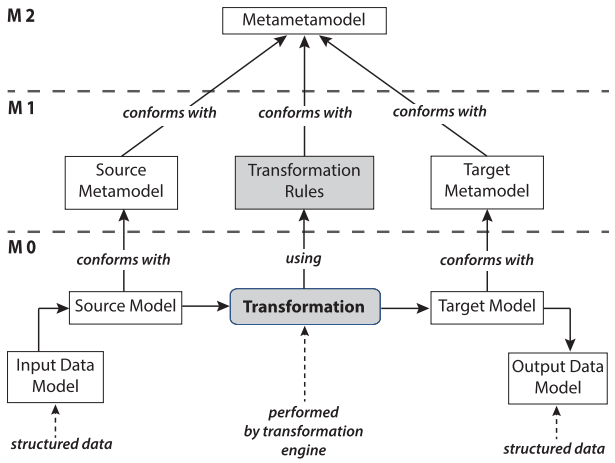


FIGURE 6. MDE for reconstructing data objects.

for a specified time or keep them permanently. The policy is supported by DDS.

- *Data Reliability:* This policy specifies the reliability of data objects when they are used in the communication. It specifies how to keep the data to insure the reliable delivery. The policy is supported by protocols such as DDS and MQTT.

V. USE CASE

This section presents a use case on the proposed approach in Section IV. It describes implementing DDIL to multiple communicating applications in SG. These applications are distributed in four different domains (i.e., operation, generation, distribution, and customer). As described in Section II, the heterogeneity of the communication protocols deteriorates the realization of SG. Therefore, in this use case we demonstrate the ability of DDIL to enable seamless connectivity among applications that are built upon heterogeneous protocols and data models. We first developed four applications based on three different protocols – DDS, OPC-UA, and MQTT. We then integrate these applications with DDIL. The above protocols are used since they are commonly used in the power system.

OPC-UA is widely adopted in the power system especially in the distribution domain (e.g., distributed energy applications, monitoring/protection) [26]. Filed devices in the distribution domain (e.g., data acquisition, protection relays) communicate their data based on OPC-UA with local servers. The servers in turn communicate the data with central SCADA applications at the operation domain. However, significant data interoperability issues occur between the operation domain and the distribution domain since they consist of heterogeneous data models [11].

MQTT has been adopted in the customer domain due to its ability to efficiently disseminate data especially when used for constrained devices [23]. Smart devices (e.g., intelligent electronic device [IED], smart monitors) use MQTT to communicate with data acquisition nodes in the customer domain. Those nodes in turn communicate the data with SCADA applications at the operation domain. However, the operation domain utilizes more flexible protocols and different data models such as DDS and the Common Information Model (CIM) [27]. Consequently, the same issues that arise between the operation and the distribution domains still occur between the operation domain and the customer domain. Section V-A demonstrates the ability of DDIL to address the above interoperability issues through model transformation.

The use case consists of two sections data related, and communication related. The data related section (V-A) demonstrates the functionalities provided by these models: Data Source, Data Semantics, and Data Naming. The communication related section (V-B) demonstrates the functionalities provided by: Service Configuration, Quality of Service, and Data Integration. It also describes the simulated experiment, the layout of the simulated environment, and the data exchange scenarios.

A. DATA OBJECTS RECONSTRUCTION

Objects reconstruction is performed by the Data Source model based on the described principles in Section IV-B(6). The Ecore model is used for reconstructing data objects, which means that in this work Ecore is used as the base of the Data Source model. Ecore is an Eclipse Modeling Framework[1] for building structured data models. The algorithms that are used for reconstructing data objects are described by the Data Semantics model. They are developed based on the Query/View/Transformation (QVT) language.[2] QVT is a model transformation language defined by the Object Management Group (OMG). The Ecore model and the QVT language are chosen since they are compatible. Additionally, because they are supported by a set of development tools that we leveraged in this work for developing DDIL. Given that, two data object reconstruction cases are presented. The first demonstrates reconstructing an OPC-UA data object into a DDS topic, while the second demonstrates reconstructing an OPC-UA data object into an MQTT topic.

1) RECONSTRUCTING OPC-UA OBJECT INTO A DDS TOPIC

This object reconstruction case demonstrates transforming MMTR (Metering) which is an OPC-UA data object into a DDS topic. MMTR is a logical node that represents a metering functionality of a device in the electric power system. It consists of set of grouped values. The information provided by MMTR is used for several purposes but mainly for billing. The abstract data model of MMTR is defined based on IEC 61850 [39] which is a standard application protocol for electrical substations.

Figure 7 shows the Ecore representation of MMTR. In the figure, the Data Model object represents the root object defined by the Data Source model. All children objects inherent the properties of the Data Model. The MMTR Object and

[1] www.eclipse.org/modeling/emf/
[2] www.omg.org/spec/QVT/

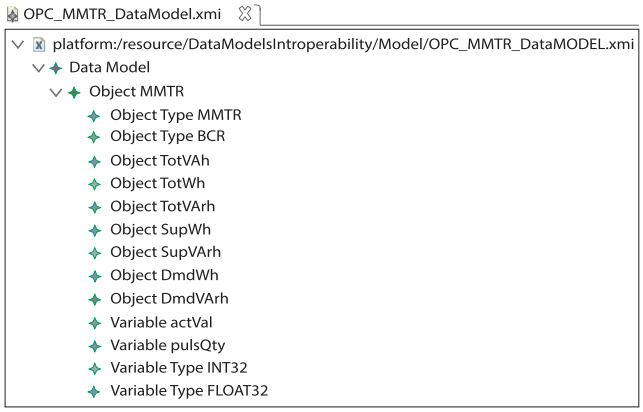


FIGURE 7. Ecore model of OPC-UA *MMTR object*.

ObjectType represent the type, properties, and structure of the MMTR logical node. The BCR (Binary Counter Reading) ObjectType defines the type of attributes (i.e., actVal, pulsQty) that are used to describe the MMTR object. MMTR consists of seven objects which are TotVAh, TotWh, TotVArh, SupWh, SupVArh, DmdWh, and DmdVArh. These objects represent properties such as apparent energy, real energy supply, and real energy demand.

In the reconstruction process, MMTR ObjectType is transformed into a UML class with a set of properties. The properties of the MMTR Object are also transformed into UML classes representing the original content that is defined by the OPC-UA data object. A DDS topic is created using the reconstructed properties. Figure 8 shows the QVT generated trace which results from transforming OPC-UA MMTR object into a DDS topic. In this work, the auto-generated trace is used to validate the correctness of the transformation process. DDS defines each topic uniquely by a topic name and a type [13]. Therefore, two properties (topic_name and type_name) are generated and attached to the reconstructed topic as shown in the figure. The properties of MMTR are represented as set of attributes by the new topic structure.

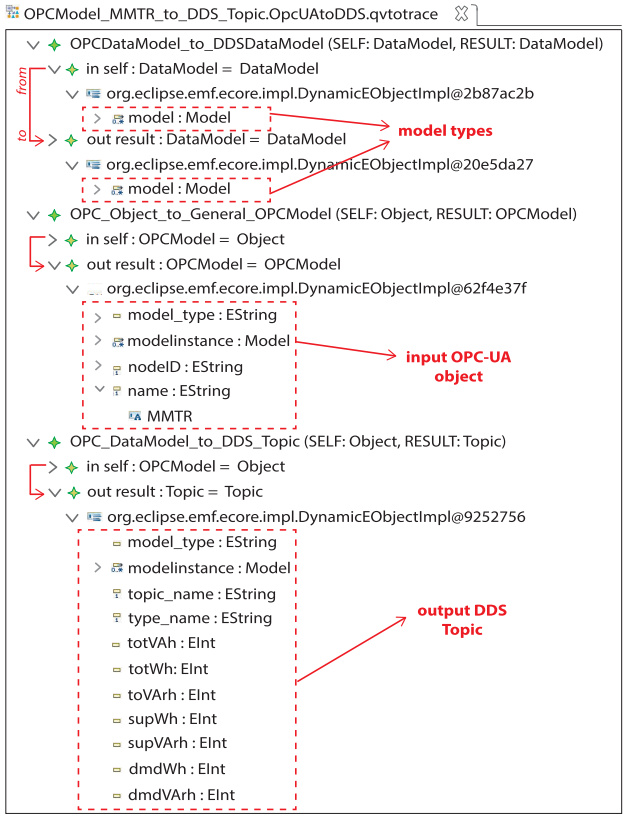


FIGURE 8. Trace of transforming OPC-UA object into DDS topic.

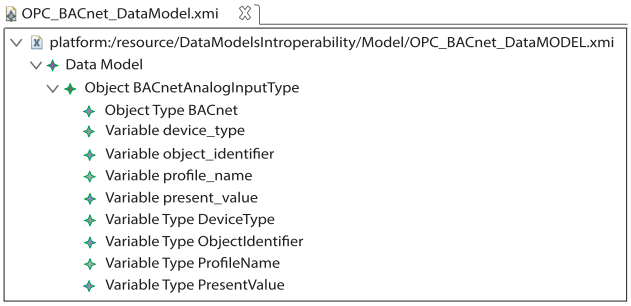


FIGURE 9. Ecore model of OPC-UA *BACnetAnalog- InputType* object.

2) RECONSTRUCTING OPC-UA OBJECT INTO AN MQTT TOPIC

This object reconstruction case demonstrates transforming BACnetAnalogInputType OPC-UA object into an MQTT topic. This object is based on the BACnet standard which is a protocol for building automation systems [40]. The BACnetAnalogInputType Object is used to represent sensing data such as temperature readings and faults data. BACnetAnalogInputType consists of several variables and types as follows: device_type, object_identifier, profile_name, present_value. They represent the actual reading values as well as the device information. Figure 9 depicts the Ecore model of the OPCU-UA BACnetAnalogInputType object.

The BACnetAnalogInputType Object is transformed into a UML class with a set of properties. These properties are used to generate an MQTT topic. Each MQTT topic is uniquely identified within a broker. Therefore, a (topicID) is generated and attached to the reconstructed topic. Figure 10 shows the QVT auto-generated trace of transforming the OPC-UA BACnetAnalogInputType object into an MQTT topic.

B. APPLYING DDIL FOR CONNECTIVITY

Given the above cases of objects reconstruction, the following demonstrates applying DDIL for enabling inter-domain connectivity in SG. SG requires all power domains to be connected to the operation domain [41]. This is because the operation domain is responsible of managing major power functionalities such as maintenance, billing, and protection. However, this connectivity is still not achievable due to the challenges mentioned in Section II. DDIL aims at filling the gap by allowing different applications to communicate even if they are developed upon different communication protocols.

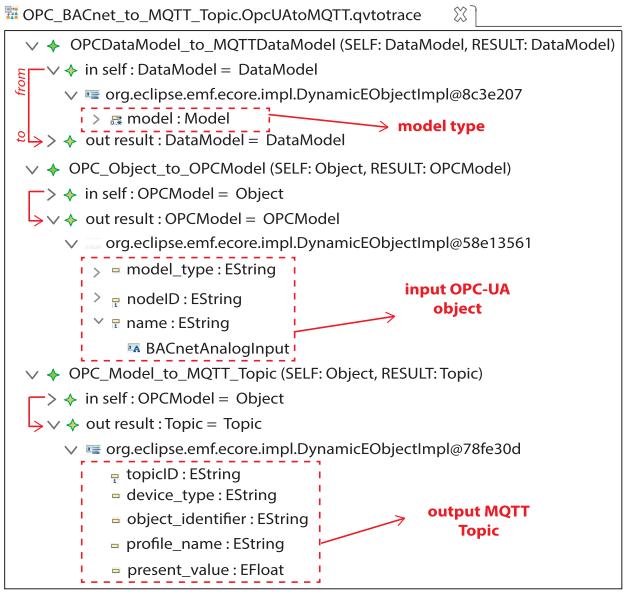


FIGURE 10. Trace of transforming OPC-UA object into MQTT topic.

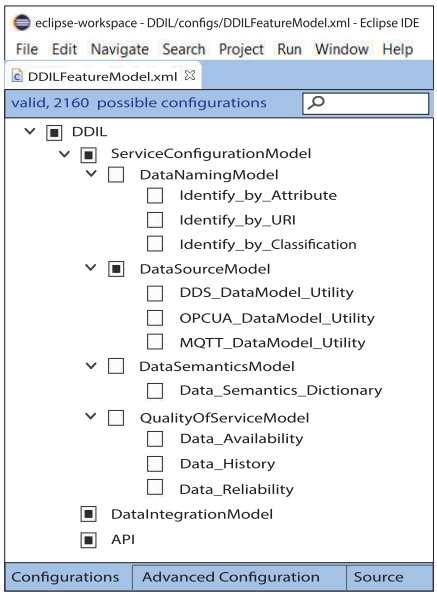


FIGURE 11. Representation of DDIL features by *FeatureIDE*.

The following environment is built to simulate the interaction between four different power domains (i.e., operation, customer, distribution, generation) within SG. The data is exchanged among four different devices. In the operation domain, we simulated a distribution server which is a PC with augmented computing capabilities. In the customer domain, we simulated a control computer, such as an automation control computer (ACC) which is used to send and receive control commands. In the distribution domain, we simulated an OPC-UA server which is used for collecting data from field devices. In the generation domain, we simulated a SCADA computer which is typically used for automation and protection.

The assumption in the following experiments is that all SG domains are connected to each other in terms of networks configuration. For instance, if an application in the operation domain requires data to be retrieved from the distribution, then the operation domain application is assumed to be connected the distribution domain application. Another assumption is that the devices that participate in the communication (e.g., computers, embedded devices) support DDIL in terms of operating systems, data models, and software models. Finally, DDIL is assumed to be installed in a distributed manner. This means that DDIL might be tailored to suit an application's requirements. Given that, we have developed the simulated testbed based on the above profiled models. We implemented the system using Java with the Ecore model. We also implemented an API layer using Java to ease integrating the developed applications with DDIL. Throughout the experiment, the exchanged/transformed data has been stored in simple file system to provide data storage and retrieval.

The configuration of applications is provided through the Service Configuration model. For performing the configuration, we use FeatureIDE [42]. FeatureIDE is an Eclipse plug-in tool for enabling feature-oriented software development. Figure 11 shows the implementation of DDIL features in FeatureIDE. The features are developed based on the feature model in Section IV-B(2). The features under DataNamingModel are used for identifying data objects. The features under DataSourceModel are used to support data models of three different protocols: DDS, OPC-UA, and MQTT. An application can configure one or more of these data models. The Data_Semantics_Dictionary feature under DataSemanticsModel provides a semantics dictionary to guide the process of objects reconstruction. The features under QualityOfServiceModel are used to configure quality requirements in the underlying protocol. Based on the configuration of each application, FeatureIDE generates a set of Java classes which can be then compiled and linked with the application through the API model.

In the use case, DDIL is configured to four different applications in four different power domains as follows: 1) For a SCADA application in generation domain, DDIL is configured to support DDS. 2) For a server application in distribution domain, DDIL supports OPC-UA. 3) For a control application in customer domain, DDIL supports MQTT. 4) For a connectivity application in operation domain, DDIL is configured to support all the communication protocols in the other three domains. Hence, this application acts as a connectivity hub among the domains. This means that the connectivity application consists of three integration proxies to facilitate integrating data from DDS, OPC-UA, and MQTT.

Figure 12 presents a running example of the above four configured applications. It shows two sets of communication scenarios. The first set is labeled by steps (1, 2, and 3). It outlines data exchange that occurs between connectivity application, OPC-UA control server, and SCADA application. In SG, power is generated and distributed only when

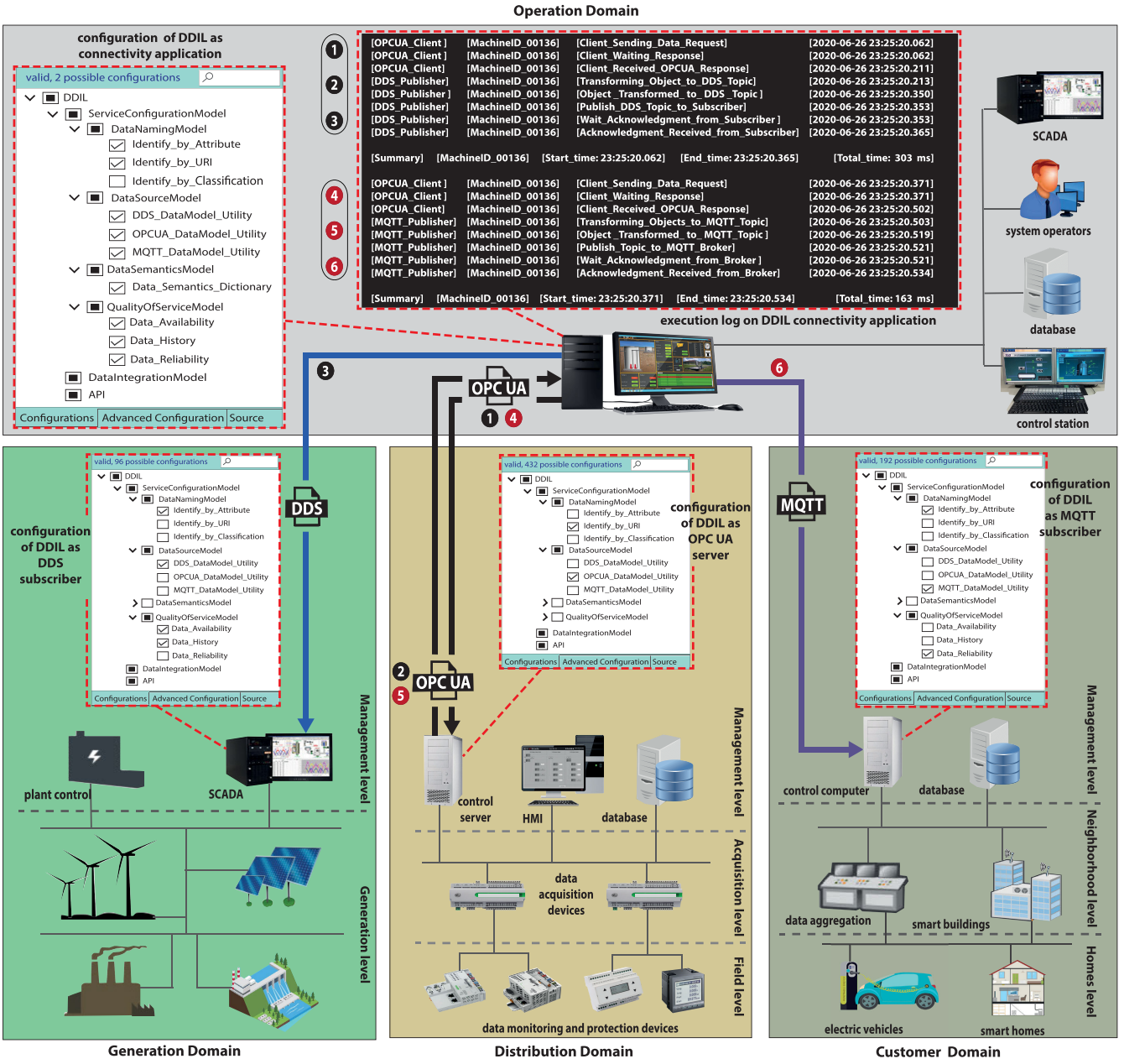


FIGURE 12. Running DDIL-based applications for enabling inter-domain connectivity in SG.

there is demand [43]. Therefore, a SCADA application in the generation domain should receive demand information from an application at the distribution domain. However, these applications utilize heterogeneous data models. The SCADA application uses DDS data model, while the control server application uses OPC-UA data model. The connectivity application is used to bridge the gap between these applications. Data exchange occurs as follows: first (step 1), the connectivity application requests data from the application at the OPC-UA control server. The server then (step 2) sends the reply to the connectivity application. In the final step (3), the connectivity application reconstructs the data object from OPC-UA into a DDS topic and sends it to the SCADA application in the generation domain. The reconstruction of the data follows the approach shown in Section V-A(1). The figure shows the execution log on the connectivity application. The top section describes the steps (1, 2, and 3). It shows participant, machine ID, performed action, timestamp of each action, and summary of time.

The second set of data exchange scenario is labeled by the steps (4, 5, and 6). It describes communication that occurs between connectivity application, OPC-UA control server, and control computer. In SG, the customer domain makes control decisions (e.g., power reroute, protective

action) based on information received from the distribution domain [44]. Therefore, in this scenario, first (step 4) the connectivity application requests data from the application at the OPC-UA control server in the distribution domain. Then, response is sent back (step 5) from the control server. Finally, the connectivity application reconstructs OPC-UA data object into an MQTT topic and sends it to the application at the control computer (step 6) at the customer domain. The object reconstruction occurs based on the example described in Section V-A(2). The bottom section of the execution log in figure describes the steps (4, 5, and 6).

VI. EVALUATION

The ideal environment for testing DDIL is a real-world SG system. However, SG is still in early development stages as described by [45]. Additionally, there is no fully developed SG system. It is also difficult to utilize modern power grids since they belong to the private sectors. Therefore, we developed the simulated environment based on the connectivity requirements of NIST [21] and the U.S. Department of Energy [45]. First, we comprehensively reviewed the relationships between the communicating components in the power system. We studied communication functionalities, paradigms, and data flow scenarios. We then developed the layout to the environment as shown by Figure 12. Based on the running examples of the four above applications, it can be seen that DDIL has the capabilities of supporting seamless connectivity in SG domains even when heterogeneous communication technologies are used. The following describe the major advantages of DDIL which can significantly improve connectivity in SG:

- The modularity of DDIL design has led to addressing the interoperability issues at different levels. For example, data interoperability issues are tackled by the Data Source model. On the other hand, protocol interoperability issues are tackled by the Data Integration model. This separation of concerns allows for managing connectivity challenges in more productive manners.
- The design of DDIL promotes rapid developments of SG applications. This is because DDIL allows developers to focus on building applications rather than addressing data interoperability and connectivity issues. For instance, the provision of the API model allows developers to build their applications in a different context and once they are fully developed, they can just integrate them with DDIL.
- The design of DDIL promotes utilizing pre-existing technologies. For instance, in this work, we used Ecore as the base of the Data Source model. We also used QVT as the base of the Data Semantics model. Furthermore, we implemented the Service Configuration model using FeatureIDE.
- The Service Configuration model allows DDIL to be tailored based on the requirements of each application. This provides flexibility which is one of the major requirements in IoT applications.

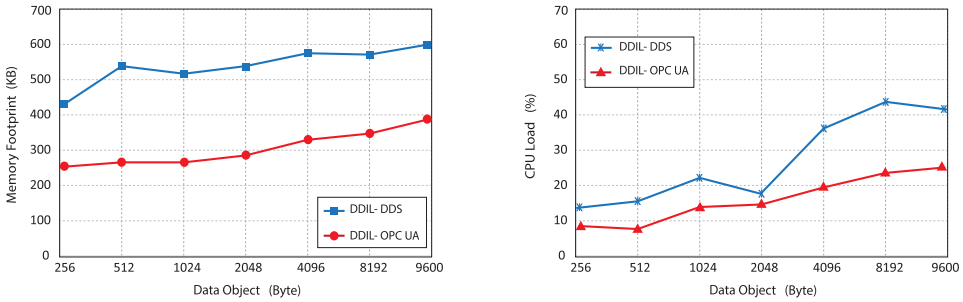
TABLE 2. Comparison between DDIL and related work.

	Lysis [28]	SAT-IoT [32]	Dist. Infrastructure for SG [29]	DDIL
Modular Design	✓	x	✓	✓
Backward Compatibility	x	x	✓	✓
Distributed	✓	x	✓	✓
Configurability	x	x	x	✓
General Purpose	x	✓	✓	✓
Utilize Existing Technologies	✓	x	x	✓
QoS Control	x	x	x	✓

Table 2 provides an evaluation view of DDIL compared to the related work described in Section III. The evaluation criterions are illustrated in the left column. The Modular Design specifies whether the proposed work supports modularity and separation of concern. Modularity is evaluated by comparing if the platform is proposed as several modules or it is introduced as one building block. The Backward Compatibility indicates if the proposed solution considers legacy systems. In this work, we evaluated this criterion by studying the exchanged data in SG. We observed that our proposed approach deals with existing data as they are represented by their original application without the need for modification. Additionally, we implement DDIL based on an application that utilizes a legacy protocol which is OPC-UA. The third criterion Distributed, specifies if the proposal is designed to be distributed on different applications or centralized on specific application. The Configurability indicates if the proposal provides set of configurable features to accommodate applications' requirements. Configurability in this work is enabled since the work is designed based on FOM. The General Purpose specifies whether the model is generic or specific to a certain IoT application. Utilize Existing Technologies indicates whether if it is possible to utilize existing technologies in developing the proposed solution. We evaluated this criterion by utilizing existing technologies (e.g., Ecore, QVT) in building DDIL. Finally, the QoS Support specifies if the quality requirements are inherently considered in the design of the proposed work.

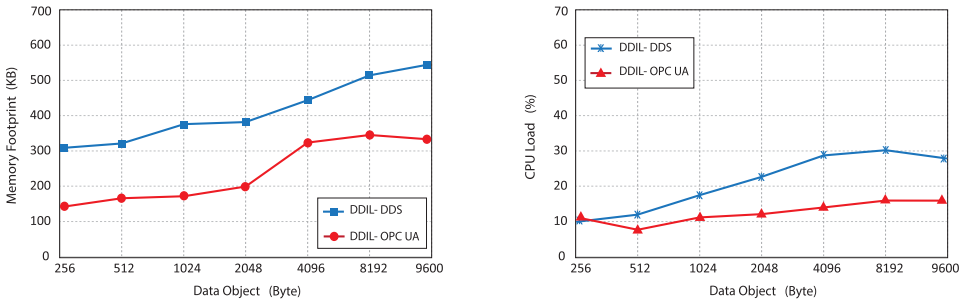
The proposed approach in this work has been tested on five different devices with different computing capabilities (i.e., memory, CPU). Those devices represent equipment that are currently used in power grids. Table 3 shows the simulated devices where the Category column categorises the devices based on their hardware resources. In the experiments, we use deferent data object size ranging from 256 bytes up to 9600 bytes. These objects are chosen since they are the typical data size that are exchanged within power domains [14], [46]. For each device in the table, the experiment is repeated 100 times. This means that each device sends/receives 100 messages. Then, the performance of DDIL is observed on both capable devices as well as constrained devices.

We observed that DDIL performs well when implemented with capable devices. For example, we measure the utilized memory and CPU load when DDIL is implemented on ACC



(a) Memory Usage (b) CPU Usage

FIGURE 13. Resources utilization on Raspberry Pi.



(a) Memory Usage (b) CPU Usage

FIGURE 14. Resources utilization on Raspberry Pi2.

TABLE 3. Utilized devices and their capabilities.

Device Name	Hardware Example	CPU (GHz)	Memory (GB)	Category
Automation Control Computer (ACC)	AIS-Q572	3.3	16	Capable
Data Concentrator Unit (DCU)	SEL-3573	3.1	8	Capable
Automation Industrial Computer (AIC)	IPC-4	2.16	4	Capable
Wireless Energy Monitor (WEM)	Raspberry Pi2	0.9	1	Constrained
Wireless Sensor Node (WSN)	Raspberry Pi	0.7	0.5	Constrained

which has 3.3 GHz CPU power and 16 GB of memory. We notice that the resource consumption was trivial. Particularly, with the largest object size of 9600 bytes, the memory consumption does not exceed 0.0003% of the device memory, and the CPU load is less than 2.8% even with the largest object size. Similar results, are observed with DCU which has 3.1 GHz CPU power and 8 GB of memory and AIC which has 2.16 GHz CPU power and 4 GB of memory. In fact, with all capable devices represented in the table, the utilized memory does not exceed 0.0017% and the CPU load is less than 6.2% even with the largest object size of 9600 bytes.

To further test the performance of the proposed approach, we implemented DDIL on the constrained devices in Table 3. Similar to the above experiments, we measure the performance of DDIL on these devices in terms of memory footprint and CPU load. In the performance tests, DDIL is configured to communicate based on either DDS or OPC-UA. We do not consider MQTT since it is a lightweight protocol and typically does not deplete the computing resources in a device and that what we observed in this experiment.

For the first device WSN, DDIL is configured twice. First, DDIL is configured to communicate based on DDS to simulate a sensor that sends protection data in the distribution domain. For the second configuration, DDIL is set to communicate based on OPC-UA to simulate an energy monitor in the customer domain. The performance of DDIL is measured based on these two configurations. Figure 13 graph (a) shows the memory footprint based on DDIL-DDS and DDIL-OPC UA. It is seen that DDIL-DDS consumes more memory than DDIL-OPC UA. This is expected due to the nature of DDS since it requires more computing resources. With the largest data object of 9600-bytes, DDIL-DDS consumes 607 KB. Compare to DDIL-DDS, DDIL-OPC UA consumes only 397 KB for the same data object size. In both configurations, the memory consumption does not exceed 0.05% of the device memory which is 0.5 GB. The graph (b) shows the CPU load of DDIL-DDS and DDIL-OPC UA. Similarly, DDIL-DDS consumes more CPU than DDIL-OPC UA. With DDIL-DDS, the highest CPU load with data object 9600-bytes is 41%, while with DDIL-OPC UA the highest CPU load for same data object size is only 24%.

Figure 14 shows the performance of DDIL with WEM. Similar to the above experiment, with DDIL-DDS, the device is used to simulate a sensor device in the distribution domain,

and with DDIL-OPC UA, the device is used to simulate an energy monitor in the customer domain. The graph (a) shows the memory footprint. With DDIL-DDS, the memory footprint with the largest data object of 9600-bytes is 543 KB. On the other hand, DDIL-OPC UA, utilizes only 332 for the same data object size. The graph (b) shows the CPU load. It shows that with DDIL-DDS, the highest CPU load with data object-9600 byte is 28%. Compare to that, DDIL-OPC UA consumes only 16% for the same object size.

Based on the above experiments, we observe that communication protocols are inherently different and each protocol requires certain computing capabilities. We also have concluded that the configurability of a communication platform greatly improves the communication since both application and device requirements are considered. Overall, we observe that DDIL has the capabilities to address the interoperability issues that arise from the heterogeneity of the communication protocols with acceptable performance.

A. CONNECTIVITY CONSIDERATION AND CHALLENGES

There are several challenges in adopting any connectivity solution for IoT applications. Those challenges are due to factors such as heterogeneity of data models, legacy systems integration, and diversity of venders and service providers. The following lists the challenges and the requirements that should be addressed in the context of adopting DDIL to IoT applications:

(a) Lack of knowledge of communication scenarios among applications in IoT domains. In order to adopt DDIL, the communication scenarios among applications need to be well defined. This is because understanding the communicating components and their communications is a prerequisite for configuring DDIL.

(b) Legacy systems lack of flexibility. The majority of legacy applications were developed with little consideration for systems integration, which means they were not designed to be integrated with new technologies. Therefore, identifying the integration requirements is a key factor for adopting DDIL.

(c) Comprehensive understanding of utilized data models. Due to the diversity of IoT data models, it is essential to understand the structure of each data model. This is because constructing and reconstructing data objects is the core of any connectivity model including DDIL.

(d) Comprehensive knowledge of functional requirements in IoT applications. This is required to support proper development of DDIL models.

(e) Comprehensive knowledge of quality requirements in IoT applications. Different IoT applications have different quality requirements. Therefore, it is necessary to understand what the non-functional requirements are prior to adopting DDIL in any IoT application.

VII. CONCLUSION

In this work, we have presented DDIL which is a connectivity layer that allows applications in IoT domains to seamlessly exchange data. The layer aims at addressing interoperability issues that occur due to the heterogeneity of data models of the communication protocols. DDIL is constructed based on two software development approaches (i.e., MDE, FOM). It addresses data interoperability issues through model transformation techniques. DDIL is developed into set of configurable features to support the flexibility requirements in IoT applications. The viability of DDIL is tested in a simulated smart grid environment. In this environment, four applications are built upon three heterogeneous protocols. The performance of the proposed approach has been tested on two constrained devices. The results show that DDIL is fixable to be implemented even on constrained devices. Overall, the results prove that DDIL enables the communicating applications to seamlessly exchange data. In the future work, we plan to extend DDIL to include application layer protocols such as IEC 61850 and CIM. Additionally, there is a plan to test the approach in other IoT applications such as the e-health system. We also need to expand the QoS requirements to include not only quality of data but also quality of communication to accommodate different applications requirements.

REFERENCES

[1] A. Ngu, M. Gutierrez, V. Metsis, S. Nepal, and Q. Sheng, "IoT middleware: A survey on issues and enabling technologies," *IEEE Internet Things J.*, vol. 4, no. 1, pp. 1–20, Feb. 2017.

[2] S. Andreev, O. Galinina, A. Pyattaev, M. Gerasimenko, T. Tirronen, J. Torsner, J. Sachs, M. Dohler, and Y. Koucheryavy, "Understanding the IoT connectivity landscape: A contemporary M2M radio technology roadmap," *IEEE Commun. Mag.*, vol. 53, no. 9, pp. 32–40, Sep. 2015.

[3] J. Ding, M. Nemati, C. Ranaweera, and J. Choi, "IoT connectivity technologies and applications: A survey," *IEEE Access*, vol. 8, pp. 67646–67673, 2020.

[4] A. Alaerjan, D. Kim, H. Ming, and H. Kim, "Configurable DDS as uniform middleware for data communication in smart grids," *Energies*, vol. 13, no. 1839, pp. 1–29, 2020.

[5] M. Noura, M. Atiquzzaman, and M. Gaedke, "Interoperability in Internet of Things: Taxonomies and open challenges," *Mobile Netw. Appl.*, vol. 24, no. 3, pp. 796–809, Jul. 2018.

[6] O. Givehchi, K. Landsdorf, P. Simoens, and A. W. Colombo, "Interoperability for industrial cyber-physical systems: An approach for legacy systems," *IEEE Trans. Ind. Informat.*, vol. 13, no. 6, pp. 3370–3378, Dec. 2017.

[7] A. Mazayev, J. A. Martins, and N. Correia, "Interoperability in IoT through the semantic profiling of objects," *IEEE Access*, vol. 6, pp. 19379–19385, 2018.

[8] Y. Li, X. Huang, and S. Wang, "Multiple protocols interworking with open connectivity foundation in fog networks," *IEEE Access*, vol. 7, pp. 60764–60773, 2019.

[9] T. Hardjono, A. Lipton, and A. Pentland, "Toward an interoperability architecture for blockchain autonomous systems," *IEEE Trans. Eng. Manag.*, vol. 67, no. 4, pp. 1298–1309, Nov. 2020.

[10] Industrial Internet Consortium, "The industrial Internet of Things volume G5: Connectivity framework," Ind. Internet Consortium, Needham, MA, USA, Tech. Rep. PB:20170228, Feb. 2017.

[11] B. Lee and D.-K. Kim, "Harmonizing IEC 61850 and CIM for connectivity of substation automation," *Comput. Standards Interfaces*, vol. 50, pp. 199–208, Feb. 2017.

[12] J. Kim, J. Lee, J. Kim, and J. Yun, "M2M service platforms: Survey, issues, and enabling technologies," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 61–76, 1st Quart., 2014.

[13] Object Management Group, *Data Distribution Service (DDS)*, Standard 2015-04-10, 2015. [Online]. Available: http://www.omg.org

[14] A. Alaerjan, "A hybrid communication platform for supporting the interoperability in smart grids," Ph.D. dissertation, Dept. Comput. Sci. Eng., Oakland Univ., Rochester, MI, USA, Apr. 2019.

[15] J. Manyika, M. Chui, P. Bisson, J, Woetzel, R. Dobbs, J, Bughin, and D. Aharon, "The Internet of Things: Mapping the value beyond the hype," McKinsey Global Inst., New York, NY, USA, Tech. Rep. 06-01-2015, Jun. 2015.
[16] S. Sinche, D. Raposo, N. Armando, A. Rodrigues, F. Boavida, V. Pereira, and J. S. Silva, "A survey of IoT management protocols and frameworks," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 2, pp. 1168–1190, 2nd Quart., 2020.
[17] A. Eshpeter and P. Eng., "Resolving the challenges of multiple vendor 61850 implementations," in *Proc. IEEE/PES Transmiss. Distrib. Conf. Expo. (T D)*, May 2016, pp. 1–7.
[18] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A survey on enabling technologies, protocols, and applications," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 2347–2376, Jun. 2015.
[19] L. Barbierato, A. Estebsari, E. Pons, M. Pau, F. Salassa, M. Ghirardi, and E. Patti, "A distributed IoT infrastructure to test and deployreal-time demand response in smart grids," *IEEE Internet Things J.*, vol. 50, pp. 1136–1146, Feb. 2019.
[20] X. Fang, S. Misra, G. Xue, and D. Yang, "Smart grid—The new and improved power grid: A survey," *IEEE Commun. Surveys Tuts.*, vol. 14, pp. 944–980, 2012.
[21] NIST, "NIST framework and roadmap for smart grid interoperability standards, release 2.0," Nat. Inst. Standards Technol., Gaithersburg, MD, USA, Tech. Rep. 1108R2, Feb. 2012.
[22] A. Alaerjan, D.-K. Kim, H. Ming, and K. Malik, "Using DDS based on unified data model to improve interoperability of smart grids," in *Proc. IEEE Int. Conf. Smart Energy Grid Eng. (SEGE)*, Aug. 2018, pp. 110–114.
[23] P. Jamborsalamati, E. Fernandez, M. Moghimi, M. J. Hossain, A. Heidari, and J. Lu, "MQTT-based resource allocation of smart buildings for grid demand reduction considering unreliable communication links," *IEEE Syst. J.*, vol. 13, no. 3, pp. 3304–3315, Sep. 2019.
[24] R. Ma, H.-H. Chen, Y.-R. Huang, and W. Meng, "Smart grid communication: Its challenges and opportunities," *IEEE Trans. Smart Grid*, vol. 4, no. 1, pp. 36–46, Mar. 2013.
[25] B. Teixeira, G. Santos, T. Pinto, Z. Vale, and J. M. Corchado, "Application ontology for multi-agent and web-services' co-simulation in power and energy systems," *IEEE Access*, vol. 8, pp. 81129–81141, 2020.
[26] D.-K. Kim, B. Lee, S. Kim, H. Yang, H. Jang, D. Hong, and H. Falk, "QVT-based model transformation to support unification of IEC 61850 and IEC 61970," *IEEE Trans. Power Del.*, vol. 29, no. 2, pp. 598–606, Apr. 2014.
[27] D.-K. Kim, A. Alaerjan, L. Lu, H. Yang, and H. Jang, "Toward interoperability of smart grids," *IEEE Commun. Mag.*, vol. 55, no. 8, pp. 204–210, Aug. 2017.
[28] R. Girau, S. Martis, and L. Atzori, "Lysis: A platform for IoT distributed applications over socially connected objects," *IEEE Internet Things J.*, vol. 4, no. 1, pp. 40–51, Feb. 2017.
[29] E. Patti, A. L. A. Syrri, M. Jahn, P. Mancarella, A. Acquaviva, and E. Macii, "Distributed software infrastructure for general purpose services in smart grid," *IEEE Trans. Smart Grid*, vol. 7, no. 2, pp. 1156–1163, Mar. 2016.
[30] M. O. Farooq, I. Wheelock, and D. Pesch, "IoT-connect: An interoperability framework for smart home communication protocols," *IEEE Consum. Electron. Mag.*, vol. 9, no. 1, pp. 22–29, Jan. 2020.
[31] A. Mazayev, J. A. Martins, and N. Correia, "Interoperability in IoT through the semantic profiling of objects," *IEEE Access*, vol. 6, pp. 19379–19385, 2017.
[32] M. A. L. Pena and I. Munoz Fernandez, "SAT-IoT: An architectural model for a high-performance fog/edge/cloud IoT platform," in *Proc. IEEE 5th World Forum Internet Things (WF-IoT)*, Limerick, Ireland, Apr. 2019, pp. 633–638.
[33] S. Rajput and S. P. Singh, "Identifying Industry 4.0 IoT enablers by integrated PCA-ISM-DEMATEL approach," *IEEE Trans. Power Del.*, vol. 57, no. 8, pp. 1–34, Sep. 2019.
[34] S. K. Datta and C. Bonnet, "Next-generation, data centric and end-to-end IoT architecture based on microservices," in *Proc. IEEE Int. Conf. Consum. Electron. Asia (ICCE-Asia)*, Jeju, South Korea, Jun. 2018, pp. 1–4.
[35] K. Kang, S. Cohen, J. Hess, W. Novak, and A. Peterson, "Feature-oriented domain analysis (FODA) feasibility study," Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. CMU/SEI-90-TR-021, Nov. 1990.
[36] F. Enescu and N. Bizon, "SCADA applications for electric power system," in *Reactive Power Control in AC Power Systems* (Power Systems), N. M. Tabatabaei, A. J. Aghbolaghi, N. Bizon, and F. Blaabjerg, Eds. Cham, Switzerland: Springer, 2017, doi: 10.1007/978-3-319-51118-4_15.
[37] K. Sayed and H. Gabbar, "SCADA and smart energy grid control automation," in *Smart Energy Grid Engineering*. New York, NY, USA: Academic, Jan. 2017, pp. 481–514.
[38] Object Management Group, *OMG Unified Modeling Language*, Standard 2015-03-01, Version 2.5, 2015. [Online]. Available: http://www.omg.org
[39] *IEC 61850 Communication Networks and Systems for Power Utility Automation—Part 7-2: Basic Information and Communication Structure-Abstract Communication Service Interface (ACSI)*, document IEC 61850-7-2, 2010. [Online]. Available: http://www.iec.ch
[40] BACnet Interest Group Europe and OPC Foundation, "OPC UA information model for bacnet," Nat. Inst. Standards Technol., Gaithersburg, MD, USA, Tech. Rep. 0.17, May 2016.
[41] A. Alaerjan and D. Kim, "Adopting DDS to smart grids: Towards reliable data communication," *Commun. Comput. Inf. Sci.*, vol. 738, pp. 154–169, Apr. 2017.
[42] T. Leich, S. Apel, L. Marnitz, and G. Saake, "Tool support for feature-oriented software development: FeatureIDE: An eclipse-based approach," in *Proc. OOPSLA Workshop Eclipse Technol. eXchange (ECLIPSE)*, 2005, pp. 55–59.
[43] J. F. Martinez, J. R. Molina, P. Castillejo, and R. Diego, "Middleware architectures for the smart grid: Survey and challenges in the foreseeable future," *Energies*, vol. 6, pp. 3593–3620, Jul. 2013.
[44] J. Ekanayake, K. Liyanage, J. Wu, A. Yokoyama, and N. Jenkins, *Smart Grid Technology and Application*. Hoboken, NJ, USA: Wiley, 2012.
[45] US-Department of Energy, "Communication requirements of smart grid technologies," US-DOE, Washington, DC, USA, Tech. Rep. 10-05-2010, Oct. 2010.
[46] M. S. Almas and L. Vanfretti, "RT-HIL implementation of the hybrid synchrophasor and GOOSE-based passive islanding schemes," *IEEE Trans. Power Del.*, vol. 31, no. 3, pp. 1299–1309, Jun. 2016.

ALAA S. ALAERJAN received the B.S. degree in computer and information sciences from Jouf University, Saudi Arabia, in 2009, the M.S. degree in computer science from Ball State University, in 2013, and the Ph.D. degree in computer science and informatics from Oakland University, in 2019. He is currently an Assistant Professor with the Department of Computer Science, College of Computer and Information Sciences, Jouf University. His research interests include distributed systems, software engineering, the IoT, information security, and smart grids.