Received 5 November 2023, accepted 27 November 2023, date of publication 5 December 2023,
date of current version 13 December 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3339766

RESEARCH ARTICLE

Distributed Service Choreography Framework for Interoperability Among Prosumers and Electric Power System

PETRI KANNISTO[1,2], (Member, IEEE),
ERDEM GÜMRÜKCÜ[3], (Graduate Student Member, IEEE),
FERDINANDA PONCI[3], (Senior Member, IEEE),
ANTONELLO MONTI[3,4], (Senior Member, IEEE),
SAMI REPO[5], AND DAVID HÄSTBACKA[1], (Member, IEEE)
[1]Department of Computing Sciences, Tampere University, 33014 Tampere, Finland
[2]VDEh-Betriebsforschungsinstitut (BFI), 40237 Düsseldorf, Germany
[3]Institute for Automation of Complex Power Systems, E.ON Energy Research Center, RWTH Aachen University, 52074 Aachen, Germany
[4]Fraunhofer Institute for Applied Information Technology (FIT), 52062 Aachen, Germany
[5]Department of Electrical Engineering, Tampere University, 33014 Tampere, Finland

Corresponding author: Petri Kannisto (petri.kannisto@iki.fi)

This work was supported in part by the Projects Distributed Management of Electricity System (DisMa) through Academy of Finland under Grant 322676 and Grant 322673, in part by the Functional ICT Interoperability in Digital Energy Systems for Green Transition through Walter Ahlström Foundation in Program Tutkijat maailmalle under Grant 20220033, in part by the Flexible Energy Systems Leveraging the Optimal Integration of EVs Deployment Wave (FLOW) through Horizon Europe under Grant 101056730, and in part by the Interoperability Network for the Energy Transition (intNET) through Horizon Europe under Grant 101070086.

ABSTRACT The electric power system is transforming from a strict hierarchy into a distributed scheme that consists of prosumers and other service providers. Some services require interactions more complex than request-response. To reach functional interoperability, such services necessitate an agreement about what actions happen, in which order, and in which conditions. Such agreements can be modeled as service choreographies, a concept familiar from microservice architectures (MSA). However, the management of choreographies is demanding, especially in inter-organizational (or actor-to-actor) schemes, if changes occur frequently or the number of participants and choreography instances is high. This article proposes Discografia, a framework to support the distributed execution of service choreographies modeled with the standard Business Process Model and Notation 2.0 (BPMN2). Beyond distributed execution, certain operations must be centralized, specifically for security as well as lifecycle and execution support. The framework is proven with a software prototype to demonstrate applicability. The prototype coordinates the charging of electric vehicles (EV) among competing aggregators operating in urban charging hubs (UCH). Additionally, the concept is evaluated with theoretical calculations that suggest reduced manual effort in communicating choreography specifications. Thus, this paper argues that Discografia concept for distributed service choreographing facilitates inter-organizational interoperability, an essential factor in future power system.

INDEX TERMS Functional interoperability, business process execution, microservice architecture (MSA), service-oriented architecture (SOA), electric vehicles, Internet of Things (IoT), smart grid, green transition, Discografia.

I. INTRODUCTION

The electric power system of the future forms a complex system-of-systems (SoS) environment due to the interactions required between the various participants. SoS refers to architectures where the systemic complexity is present not

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

only inside one system but also in the mutual interaction between multiple systems [1]. The SoS in power systems originates from the interdependency of the physical, automation-related, commercial, and regulatory aspects of the entities with own goals. The entities include, for instance, power plants, Transmission System Operators (TSO) for delivery via main lines, Distribution System Operators (DSO) for local delivery, aggregators that group consumption or production and often operate in the market without any physical assets, and customers. This work considers that the "customer" is primarily the end customer, although industrial customers can take an analogous yet more powerful role. In modern schemes, the customer is often referred to as *prosumer* due to its ability to both consume and produce electricity [2]. Prosumers operate in distribution grids, forming microgrids or energy communities that sell services to other prosumers, microgrids, and operators [3]. This trend receives boost from renewable energy sources, which are leading to a widely distributed energy production while requiring flexibility services from smaller-scale actors [4]. Overall, the number of actors is arbitrary and they can have many roles, which makes the SoS complex.

Interaction within the power system necessitates interoperability, which refers to the ability of systems and organizations to cooperate thanks to data and information exchange. The various definitions of interoperability can stress devices and systems [5] or organizations [6]. This work considers that the "organization" can refer to any business actor, including a natural person that purchases or sells services. In the power system, interoperability is challenging as the actors must communicate over organizational borders and the subsystems and devices are designed by separate vendors according to specific standards and for specific functions, and not intended for the broad cooperation needs of modern power systems. On the other hand, interoperability requires continuous attention due to repeated changes in technology and the business environment, such as the spread of electric vehicles (EV) and new prosumer schemes.

To facilitate interoperability, it is possible to model multi-actor system-to-system interactions as service choreographies. A service choreography describes how a group of systems reaches a common goal by executing a collaborative workflow that defines certain pieces of interaction and therefore helps in reaching functional interoperability. The execution can occur in a distributed way instead of centralized coordination (or orchestration), because distribution enables more flexible schemes and reduces the need for manual configuration. The choreographies can be modeled in Business Process Model and Notation 2.0 (BPMN2 [7]), which is a standard, providing both a modeling technique and a graphical notation for visualization. BPMN is supported by a range of open or commercial tools.

This work proposes a service choreography framework based on the hypothesis that commonly agreed choreographies can be executed in a distributed fashion in

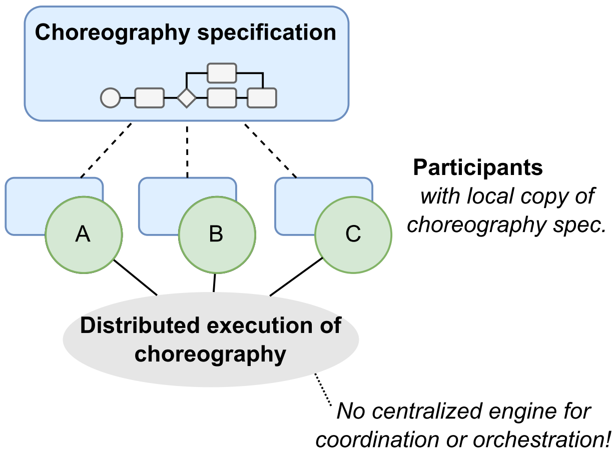
Choreography specification
Participants
with local copy of choreography spec.
A
B
C
Distributed execution of choreography
No centralized engine for coordination or orchestration!

FIGURE 1. Service choreography executed in a distributed fashion.

inter-organizational schemes, each participant taking care of its own responsibilities (see Figure 1). To enable the scheme, there should be a software system framework to support choreography management and participation. This work introduces such a framework, called *Discografia*, and demonstrates this with a software prototype. The prototype coordinates the charging of EVs in urban charging hubs (UCH), where competing aggregators provide services for customers but the whole should be coordinated by a UCH broker to both help the customer and protect the network from overloading. Thus, the research objectives are:

1) Outline the architecture of Discografia, a software system framework to communicate inter-organizational, machine-readable system-to-system and actor-to-actor choreography specifications and support the distributed execution of the choreographies.
2) Prove the framework with a prototype to execute the distributed choreography of coordinating the charging spot reservations for electric vehicles in urban charging hubs with multiple competing aggregators.
3) To indicate the potential advantage of the software framework in supporting distributed service choreographing, quantitatively estimate the reduction of manual work compared to the manual communication of choreography specifications.

Although everyday systems already execute distributed choreographies, these are typically implicit and hard-coded in the software systems and therefore lack the adaptability of choreography modeling. As an alternative, explicit choreography specifications provide an agreement of the workflow that can be implemented by any new actor joining the value network. On the other hand, the specification enables explicit version control to support evolution without conflicts. Besides, the advantages are beneficial even beyond the electricity domain, applying to modern systems-of-systems that are commonly designed as Microservice Architectures (MSA). MSAs in general encourage choreographing over centralized orchestration [8].

II. METHODOLOGY AND STRUCTURE

This work applies the research method of Design Science Research (DSR), where the idea is to design artifacts that solve or relieve real-life problems. The term artifact can refer to both design concepts and software [9]. DSR can be seen to consist of three cycles: relevance cycle to observe a real-life problem and return a solution, rigor cycle to both apply scientific knowledge and contribute to it, and finally, design cycle for the design process itself [10].

In this article, DSR is applied with the following sequence, taking iterations as necessary:

- **Section III: background.** Establish the scientific background.
- **Section IV: solution design.** Outline the architectural principles of the choreography framework.
- **Section V: proof of concept.** Indicate relevance and feasibility with a proof of concept.
- **Section VI: evaluation.** Evaluate the framework proposal and show the advantages.
- **Section VII: assessment of results.**
- **Sections VIII and IX: discussion and conclusions.**

These steps cover the three cycles as defined in [10]. Section III forms the relevance cycle by associating the scientific background. Sections IV and V provide the design cycle, producing artifacts. Finally, the rigor cycle stems from the research as a whole, contributing to scientific knowledge, but the core rigor still comes from the design of Section IV.

III. BACKGROUND OF SERVICE CHOREOGRAPHIES

The state of art of this work includes interoperability, microservices, choreography modeling, and the execution of these models. Additionally, the whole is motivated by the electricity-domain-specific development towards more flexible value chains built upon servitization.

From the viewpoint of DSR method, this section establishes the scientific background. That is, it provides the foundation of the relevance cycle, although the full relevance can only be reached once the proposed framework is implemented in an everyday business environment.

A. SERVITIZATION IN POWER SYSTEMS

Within the power systems domain, the application of services is currently emerging to replace the conventional hierarchical, centrally managed systems. This subsection provides an overview of the progress, supported with examples.

The breakdown of the hierarchy receives support from the concepts prosumers [11] and microgrids [12] that sell or produce power in addition to the conventional, centrally controlled market actors. These efforts can offer added value, such as microgrid blackstarting in a self-configurable, distributed fashion [13]. As the new service-oriented market appears, the operators likely want to still retain the technical control, whereas the added value is generated by communication service providers that build on top of the technical layer [14].

In addition, the earlier monoliths are broken down into modular, service-based concepts, such as the platform *Service-based Open-source Grid Automation Platform for Network Operation of the Future* (SOGNO). SOGNO provides a service-oriented solution for distribution grid automation, facilitating software maintenance and relieving the vendor lock-in experienced by DSOs [15]. SOGNO has even been adopted as a project of Linux Foundation Energy [16]. Later, SOGNO has been extended with electricity-market-related aspects to widen the original scope [17].

SOGNO has later been implemented into Platone framework. This aims is to involve actors from any level, such as TSOs, DSOs, aggregators, and customers. Platone applies distributed ledgers to reach trust between the participants. The use cases include but are not limited to voltage management, congestion management, state estimation, and energy management [18].

Servitization helps the data access from Internet of Things (IoT) devices once suitable technologies are applied. Such technologies can be, for instance, FIWARE to collect and distribute data as well as Message Queueing Telemetry Transport (MQTT) to route and deliver the data. This can facilitate the realization of services, such as energy management, service restoration, network configuration, and monitoring [19].

The service paradigm is more abstract and therefore more complex compared to the earlier operational technologies, but it is necessary to reach convergence between information technologies and operational technologies (IT/OT). The complexity can be problematic in control or protection activities that require deterministic, millisecond-grade response times. As reviewed in [20], IT/OT convergence can be contributed to by defining the functionality in software rather than hardware. According to the study, the goal is currently hindered by difficulties in the transition, including interoperability and stakeholder readiness.

Servitization does not limit to electricity but spans over the whole energy system. In this effort, Multi Energy Semantic Platform (MESP) suggests a platform for interoperability, building upon standardized information models and a message broker. Furthermore, MESP proposes that the services should be containerized for easier management, which is a common approach in microservices [21].

As the aforementioned examples have shown, various service concepts and platforms have emerged for the power system, but choreographing remains absent. Servitization reaches actors in any role, and IoT and hardware are involved too. Thus, service interfaces and networks are becoming the state-of-art method for all communication. However, no references have been discovered regarding *functional interoperability through choreographing*. It can be argued that choreographing is a higher-level problem, waiting to build upon the lower-level solutions that are currently appearing.

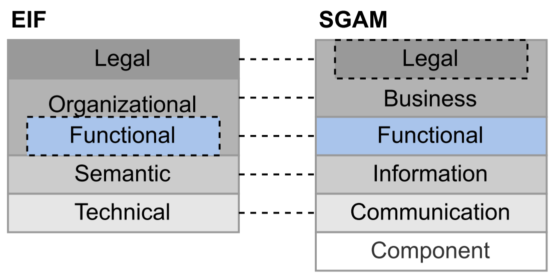

FIGURE 2. **Interoperability layers after EIF and SGAM along with implicit layers surrounded by a dashed border. Synthesized from [6] and [22].**

B. FOUNDATIONS OF CHOREOGRAPHING

Service choreographies reside in a layer called functional interoperability in interoperability models. Multiple models present a layered approach, and among these, two are considered the most important in this work: *European Interoperability Framework* (EIF) and *Smart Grid Architecture Model* (SGAM). EIF covers not only technical and semantic but also organizational and legal interoperability and remarks that all-pervasive governance is necessary [6]. EIF is domain-agnostic but useful due to its emphasis on organizations. SGAM was modeled especially for power systems and has three dimensions: interoperability layers from component to business, zones from process to market, and domains from generation to customers [22]. Figure 2 shows how the layers of EIF and SGAM align with each other. Functional interoperability, i.e., where choreographies are located, is omitted in EIF but can considered an implicit part of organizational interoperability. Respectively, SGAM lacks legal interoperability as a layer but considers this to reside in business interoperability. It can be argued that the functional layer is particularly difficult, because it should specify what activities are expected and with which participants, including special cases and the related conditions. It is more straightforward to agree about communication protocols and syntaxes (technical interoperability) or to follow certain legislation (legal interoperability) than to encounter the variety in the workflows of commercial actors. On the other hand, semantics and business aspects bring heterogeneity as well but are out of scope in this work.

Service choreographies resemble Multi-agent Systems (MAS) as both refer to reaching goals with collaborative actions, but the paradigms have a different viewpoint. A choreography describes the mutual interaction of parties, especially what messages and in which situation each should exchange to enable a collaborative process to reach its goals [23]. The MAS is concerned with the environment of operation, the actions the agents can perform to affect the environment, and the individual goals and decision making of the agents [24]. That is, in the scope of the interaction, a service choreography describes a business process with a predefined contract, whereas MAS assumes a world of possibilities. Although an agent can agree to join a choreography, the modeling of the business process is not the primary focus in MAS. Thus, unlike MAS, choreographing suits for scenarios where a predefined agreement is necessary to reach the common goal and recognize the potential outcomes and status of the workflow. On the other hand, there are suggestions for choreographies that adapt to emerging needs (e.g., [25]), which introduces strengths similar to MAS.

In microservices, choreographies typically take place in an event-driven, asynchronous manner [26, p. 192]. Cloud-based choreographies open opportunities for parallelization through the paradigm Function as a Service (FaaS), but this adds difficulty to the control of execution [27]. Regarding choreography-based systems, a concrete example is the architecture *Coordinating Optimisation of Complex Industrial Processes* (COCOP), where service-oriented software systems communicate via a message broker, executing collaborative workflows implemented as services rather than being centrally orchestrated [28]. The platform *Simulation Environment of Complex Energy System* (SimCES) follows a similar approach to simulate power systems and markets [29]. However, both COCOP and SimCES lack the tools to monitor or control choreography execution and rather rely on logs in case of problems. It can be argued that this is the usual case in state-of-art MSAs, but the limitations could be relieved with choreography-execution-related tools.

C. MODELING WITH BPMN

As the choreography specification method, this work selects *collaboration* diagrams and models from BPMN 2.0, i.e., the most recent version [7]. Among the modeling techniques of BPMN, not only collaboration but also *choreography* diagrams can describe choreographies between service-oriented systems. Collaboration diagrams can lead to ambiguous interpretations due to the lack of formal semantics [30]. Therefore, the selected modeling method could be complemented in the future, because ambiguity is a potential problem regarding functional interoperability, the goal of Discografia.

Besides the diagram notation, BPMN provides an underlying modeling technique. Because BPMN is an open standard, the models enable the service choreographies to be interpreted by software regardless of the vendor. On the other hand, the choreographies can as well be executed in a business process engine, such as the Java-based Camunda [31], Python-based SpiffWorkflow [32], or WSO2 Enterprise Integrator [33], to mention a few. Because BPMN is a vendor-independent technology and enables modeling, visualisation as well as execution, it contributes to functional interoperability and therefore the main goal of Discografia. The models are serialized as Extensible Markup Language (XML), which is widely supported among the libraries of programming languages.

BPMN has both advantages and shortcomings. The visual appearance of BPMN increases the expressiveness of workflow descriptions but only if the readers are educated for BPMN [34]. An evaluation of BPMN has shown shortcomings regarding choreography modeling. The most significant issues are the limitations of graphical expression, lack of understandability of the metamodel due to the focus on technical questions, the insufficient definition of the semantics of the choreography diagram, and the absence of a tool to trace choreographies and technical services [35]. Despite a few open issues, BPMN is considered the best modeling tool for Discografia due to its combination of expressiveness and machine-readable, standardized modeling.

D. CHOREOGRAPHY-RELATED CONCEPTS

Not all choreography-related concepts are clear without an explanation, and some concepts of this work lack an established meaning among choreographers. The later parts of this work builds upon the following definitions.

- **Choreography specification** is a computationally interpretable and executable description of a collaborative workflow of business parties, containing activities, conditions, and events
- **Choreography release** is either the first publication of a *choreography specification* or an update to an existing one
- **Choreography instance** is a joint effort of business parties to execute a *choreography specification* with computational systems
- **Choreography role** is a part of *choreography specification*, specifying what tasks a business party must fulfil and in which situation (''partnerRole'' in BPMN [7])
- **Choreography participant** executes a *choreography role* in a *choreography instance* (the concept appears with no restriction to instances in BPMN [7])
- **Choreography specifier** creates a *choreography specification* and communicates this to any business parties interested to become a *choreography participant*
- **Choreography implementer** develops and maintains software to become a *choreography participant* in *choreography instances*

E. REQUIREMENTS FOR DISTRIBUTED CHOREOGRAPHING

Discografia aims to enable and support distributed service choreographing, which has multiple requirements to fulfil. The following requirements summarize the foundation of Discografia design.

1) DISTRIBUTED CHOREOGRAPHY EXECUTION

There must not be a single engine for the execution. Instead, the environment must enable the execution to span over multiple participants that are interested only in their own choreography role. Therefore, the execution of choreographies should scale to any number of participants. In practice, this necessitates an at-least-*de-facto*-standardized method for choreography modeling. Such a method provides a common language, enabling the implementers to interpret the choreography specifications.

2) INTER-ORGANIZATIONAL CHOREOGRAPHIES

The environment must target at choreography execution that spans over organizational borders (where ''organization'' can refer to any business actor, including a natural person). Otherwise, organizations and actors cannot use the choreographies as a tool for collaboration and interoperability.

3) COMMON INTER-ORGANIZATIONAL EXECUTION LANGUAGE

As choreographies are executed, interoperability is possible only if they are communicated between organizations in a commonly used format. The format can be BPMN but any other commonly used language as well, including the common smart contract languages in the case of distributed ledgers (such as Solidity[1] or Vyper[2]).

4) SUPPORTING SERVICES

There must be a service framework to support execution in a distributed manner. First, the framework must enable the participants to distribute and retrieve choreography specifications. Second, to support choreography execution, the participants should be able to authorize themselves, discover relevant services, and communicate with other participants as modeled in each choreography specification. Such a framework has multiple advantages. It enables the choreography specifications to evolve but still be supported by participants, because the specification explicitly indicates the requirements of interaction. On the other hand, discovery enables choreography instances to start between any participants (such as EV recharging regardless of geographical location and the charging operator). Finally, authorization realizes access control and trust.

5) LIGHTWEIGHT COMMUNICATION METHOD

If the use case involves constrained devices, the communication within choreography workflows should prefer a low computational overhead. This is especially important not only in IoT devices, such as wireless sensors, but also vehicles and mobile machines. The requirement can hamper especially distributed ledger technologies that necessitate cryptographical computation to validate activities through a consensus within the network, causing overhead and delays. Such computation might face no issues in cloud systems, but this work assumes that the computation can occur in constrained machines. Still, Nguyen et al. have envisioned that distributed ledgers have future potential even for industrial IoT [36], but the vision necessitates additional work for verification.

6) APPLICATION IN POWER SYSTEMS

A generic choreography framework is, in principle, applicable in any domain. However, an application in power systems is included in the criteria to indicate the research gap where

[1]https://soliditylang.org/
[2]https://docs.vyperlang.org/

TABLE 1. Summary of how BPMN-supporting choreography execution environments support the requirements of fully distributed, inter-organizational choreographies.

	Distributed choreography execution	Inter-organizational choreographies	Common inter-organizational execution language	Supporting services	Lightweight communication method	Application in power systems
Arrowhead [37]	-	-	-	-	✓	-
Blockchain extension for Camunda [38]	✓	✓	✓	-	-	-
Caterpillar [39]	✓	✓	✓	✓	-	-
ChorChain [40]	✓	✓	✓	✓	-	-
CHOReVOLUTION [41]	-	✓	-	✓	✓	-
CoPuB [42]	✓	✓	-	-	-	-
Lorikeet [43]	✓	✓	✓	-	-	-

Discografia is situated. The requirement exists to reflect to the ongoing transformation of the centrally controlled power system into a distributed, service-oriented design for more adaptability and flexibility as required by the ever-changing business environment and renewable energies.

F. RELATED WORK AND RESEARCH GAPS

Earlier, multiple works have proposed environments for service choreographing. This subsection surveys how they meet the requirements specified in Section III-E to indicate the research gap. The survey is limited to BPMN-supporting environments, because BPMN is considered the only feasible standard to support choreography modeling, visualization, and execution. Table 1 summarizes the results that are elaborated in the following paragraphs.

Arrowhead is a generic service-oriented framework with an extension for choreographing [37]. Arrowhead assumes a centralized management scheme for choreography execution. One entity should deploy the final choreography logic to executors, and the choreographies are intended to execute within a single enterprise. There is no service to distribute choreography specifications to implementers. For a strength of Arrowhead, the communication protocol can be any in principle and therefore a lightweight infrastructure is possible if only the supporting services, such as authenticator and orchestrator, have the sufficient resources. Another advantage of Arrowhead is that it can support workflows executed by external systems [44].

A blockchain extension to Camunda brings the support for distributed ledgers to a widely used workflow management system [38]. The system assumes that each participant can execute their own choreography specification, relying on smart contacts. The smart contract language is either Solidity or Vyper. However, the system lacks a service to discover other parties at runtime. Furthermore, the distributed ledger brings computational overhead.

Caterpillar builds choreography execution upon distributed ledgers and is therefore both distributed and inter organizational [39]. The smart contracts, generated for execution, are expressed in Solidity. The essential supporting services are provided, including event monitoring, choreography specification storage and distribution, and trust between the participants. However, the distributed ledger causes overhead.

ChorChain [40] exploits distributed ledgers for choreographing as the smart contracts force the correct workflow to occur. The smart contract language is Solidity. supporting services are provided for choreography publishing and distribution as well as service discovery, and the ledger infrastructure provides trust. Nevertheless, the distributed ledger increases the amount of computation during execution.

CHOReVOLUTION [41] aims to enable distributed choreographies between organizations but still assumes a centralized cloud infrastructure to coordinate the execution. The system has features for choreography modeling, storage, and instantiation. A service inventory enables discovery, whereas identity management realizes trust. The choreographed services can have Restful interfaces and are therefore considered lightweight to the clients. CHOReVOLUTION has been evaluated at least in the domains of smart mobility and tourism [41] as well as marketing and sales [45].

CoBuP [42] is another distributed-ledger-based approach. It does not execute BPMN models but converts these into a custom-made JSON format for execution, which reduces the advantage of the standard. Distributed, inter-organizational execution is the goal. However, the services focus on choreography creation and publishing, lacking discovery.

Lorikeet [43], similar to many others, applies distributed ledgers, therefore it supports distributed and inter-organizational execution. The smart contract language is Solidity. The ledgers enable a trusted environment with services for publishing and trust, but there is no discovery functionality. The overhead from ledgers is again a drawback.

Table 1 shows that a research gap remains, because no earlier work can meet all the requirements. To realize distributed, inter-organizational execution, distributed ledgers are often suggested. However, the related overhead during execution cannot necessarily be afforded in IoT or other resource-constrained schemes, such as electric mobility. Instead, Discografia suggests a simpler approach to reach trust, authenticating the connected systems and software with keys and tokens. The communication occurs via a message broker, a technology that can be designed to bring only a low overhead (e.g., Message Queue Telemetry Transport or

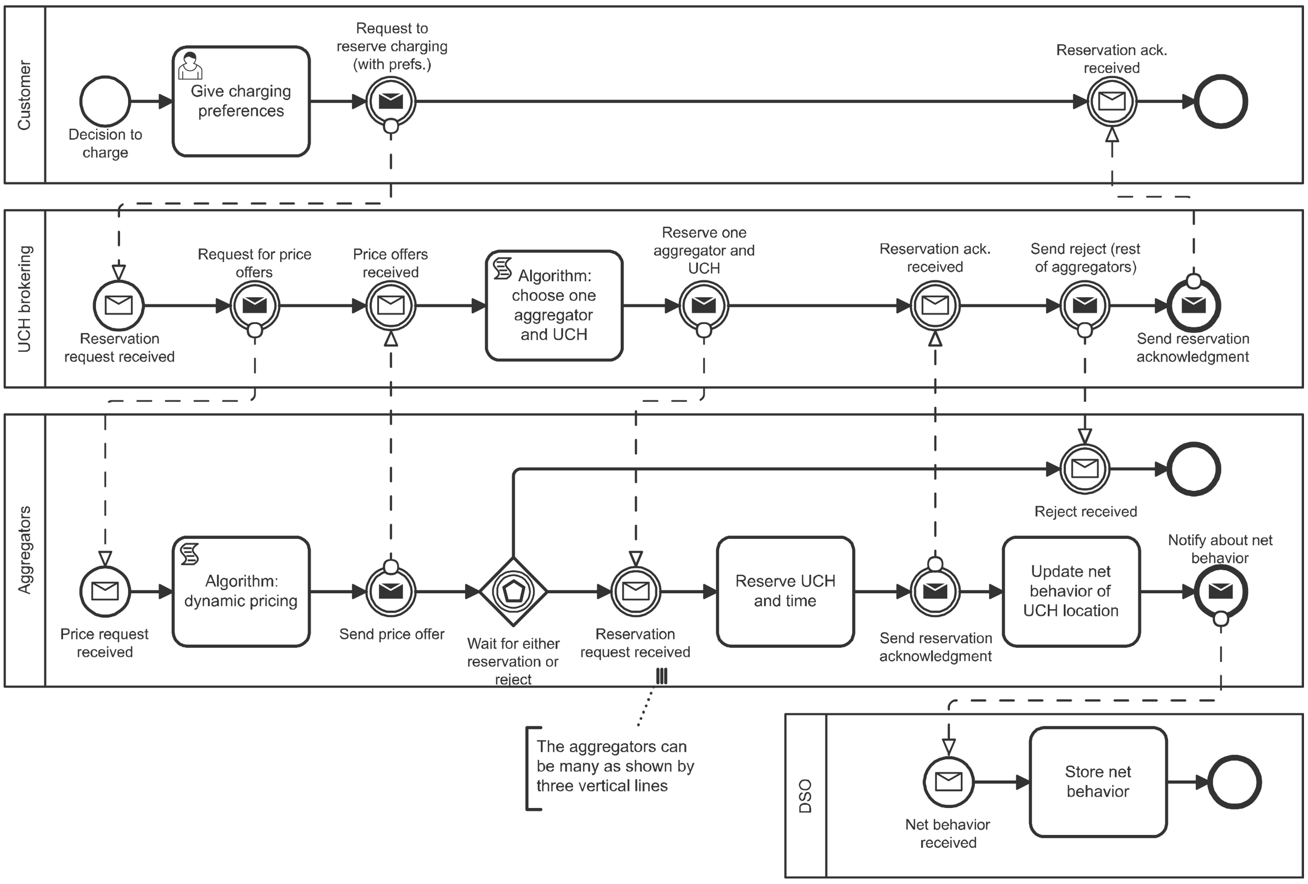
Customer
Decision to charge
Give charging preferences
Request to reserve charging (with prefs.)
Reservation ack. received
UCH brokering
Reservation request received
Request for price offers
Price offers received
Algorithm: choose one aggregator and UCH
Reserve one aggregator and UCH
Reservation ack. received
Send reject (rest of aggregators)
Send reservation acknowledgment
Aggregators
Reject received
Notify about net behavior
Price request received
Algorithm: dynamic pricing
Send price offer
Wait for either reservation or reject
Reservation request received
Reserve UCH and time
Send reservation acknowledgment
Update net behavior of UCH location
The aggregators can be many as shown by three vertical lines
DSO
Net behavior received
Store net behavior

FIGURE 3. The UCH brokering process modeled with the collaboration notation of BPMN.

MQTT [46]). This difference in the communication medium could be explored further in future research, because there can be implications to not only performance but also trust between the participants and the entire system architecture.

As a common denominator to all surveyed technologies, none has applications in power systems. In general, the literature searches returned no matches for service choreographing within the domain. Still, it can be argued that the servitization of power systems will eventually lead to the deployment and advancement of choreography management methods. These implications both introduce a clear research gap and suggest that Discografia has novelty in especially power systems.

IV. CHOREOGRAPHY FRAMEWORK DESIGN

Discografia aims to provide a framework to support choreography execution in a distributed fashion. For this aim, multiple aspects must be covered from security and the management of choreography specifications to the concrete communication of the participants. Because the framework concept is abstract, this section starts by defining a running example for a later reference.

Regarding the three-cycle DSR method, this section not only forms the design cycle but also contributes to the rigor cycle. The rigor comes from the novel scientific knowledge as well as from the application of existing knowledge in the design.

A. EXAMPLE PROCESS FROM EV CHARGING SPOT RESERVATION

For a case process, this article presents a choreography of EV charging in UCHs. It is expected that the number of EVs and therefore the volume of charging load will increase rapidly in the future. With the proliferation of EV charging infrastructure in publicly available parking lots, some urban districts can evolve into UCHs where multiple operators (or aggregators) compete for customers in the same area (as earlier proposed in [47]). In this competition, price and the availability of resources are the main market drivers, but the electricity network must still be protected from situations where the selfishness of aggregators could cause problems, such as overloading. Such protection necessitates a coordinating service, which is called UCH brokering in this use case. The brokering performs the selection of one aggregator (i.e., from multiple aggregators offering charging services) for the customer (i.e., EV driver) whenever they search for a charging service in a specific area. The selection is based on not only prices and charger-specific availability and capabilities but also on the load situation of the distribution grid.

Figure 3 illustrates the BPMN model of UCH brokering. The participants are customers, UCH brokering service, aggregators, and the local DSO. After receiving a request from the customer, the UCH brokering service asks for actual dynamic price signals from multiple aggregators operating in UCHs in the nearby area. Once the prices have been received, it chooses the best aggregator and UCH based on pricing, charging capacity, and the distance to drive to reach the UCH. The UCH broker acknowledges this choice to the selected aggregator, which reserves the required capacity for the customer and informs the DSO about the updated net behavior. Finally, the UCH broker rejects the rest of aggregators and informs the customer about the choice. The process, physically an XML document, was modeled with the online tool *BPMN.io*.[3]

The choreography includes two algorithms, namely Dynamic Pricing executed by the aggregators and Smart Routing by the UCH brokering, respectively. Dynamic Pricing is a rule-based algorithm that determines the fluctuations in the EV charging price per kWh over the estimated parking period within the scope of one aggregator. The algorithm enables the aggregator to express its preferred consumption pattern by assigning high or low prices during periods when it is necessary to reduce or increase the charging load based on existing or predicted commitments. UCH brokering utilizes these indicators in Smart Routing to determine the optimal charging profile and location for the EVs by solving an optimization problem, which covers the customer's input (e.g., the target state-of-charge or SOC at the end of the charging period) and the aggregators' price signals produced through Dynamic Pricing. This considers peak power limits (PPL) as well. The mathematical formulation of both algorithms is presented in [47].

From the perspective of service choreographing, the interaction between the UCH broker and aggregators requires most specification. These participants exchange multiple messages in a certain order. If this order is violated, the required functional interoperability will not realize, leaving one or more parties unaware of the status. To maintain interoperability, the BPMN process explicitly indicates what to do and when. On the other hand, from the viewpoint of the customer and DSO, the steps are simple.

The UCH brokering process has two purposes in this work. It provides, first, a running example for the design and, second, the case process for the prototype implementation in Section V. Still, it is notable that Discografia is a generic choreography framework and not designed for this specific use case.

B. THREE-LAYER ARCHITECTURE

Discografia specifies a three-layer framework to enable the distributed execution of service choreographies based on pre-defined choreography models (see Figure 4). The core layer is called framework layer, which is generic and supports security, the lifecycle of choreography specifications, and execution. On top of this, next comes service communication layer, which is domain specific and enables the communication between network nodes. Finally, the top layer is for

[3]https://bpmn.io/

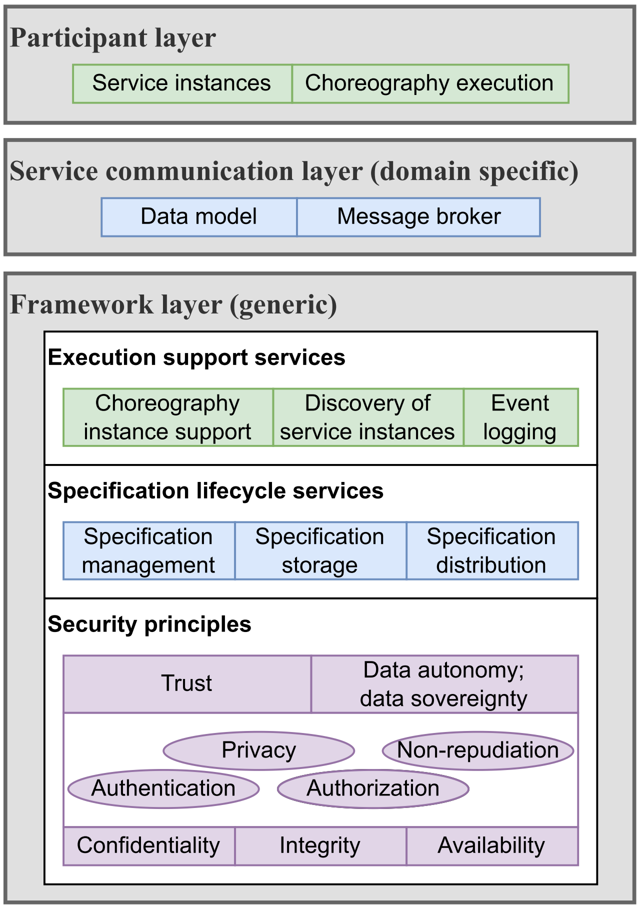

FIGURE 4. The elements of the framework.

choreography participants and the actual service instances being provided.

Although the framework concept focuses on functional interoperability, it would be incomplete without security, a communication protocol, and data model. All of these must be acknowledged, as seen in the security principles and service communication layer of Figure 4. The figure as a whole receives an elaboration in the following subsections.

C. FRAMEWORK LAYER

1) SECURITY PRINCIPLES

All modern information systems, including Discografia, must consider security in the design. Security builds upon *confidentiality*, *integrity*, and *availability* (CIA [48]). Discografia refines these further to goals, such as authentication, authorization, privacy, and non-repudiation.

In Discografia, the common security-related principles form the basis for trust and data autonomy, which bring an inter-organizational aspect to security. The service choreographies necessitate data sharing between customers and organizations. Data sharing necessitates that the business partners trust each other. Trust can be reached with data autonomy, which refers to the ability of the data owner to control who exploits the data for what and where physically. Although data-autonomy-related efforts usually consider the sharing of data sets, the concept could be extended to online data sharing too [49]. On the other hand, even data sovereignty, that is, the consideration of local laws in data processing, is important. The efforts for data autonomy and sovereignty have recently been researched related to International Data Spaces (IDS [50]) and Gaia-X [51].

In the UCH brokering example, the customer must provide their personal information, at least location and customership-related data, to the UCH broker, which will share this information with the aggregators. The customer can share information only if they can trust the other actors. Respectively, all other actors should trust each other too.

2) SPECIFICATION LIFECYCLE SERVICES

Distributed choreography execution necessitates a common version of the choreography among all participants, which is what the lifecycle services accomplish. Three aspects are considered. First, management enables the choreography specifier to publish new versions while keeping old versions available in case some services still want to support these. When executed, the participants will have to agree on the version used. Second, the various specifications and their versions must be stored somewhere. Third, the participants need a means to access the specifications, that is, there must be a means for distribution.

In the UCH brokering example, the lifecycle services enable an agreement, i.e., an up-to-date choreography specification, among the participants about the UCH brokering workflow. For these needs, the choreography specifier (whichever organization this is) utilizes the specification management and storage services of the framework. This functionality must be centralized to enable a single location for the specification although the execution itself occurs in a distributed manner. Eventually, the specification distribution service makes this available to the participants.

3) EXECUTION SUPPORT SERVICES

Although the execution of choreographies is distributed, certain execution-related support must be centralized. The goal of the framework is, however, to keep this in the bare minimum.

The execution support services are choreography instance support, the discovery of service instances, and event logging. The choreography instance support maintains a correlation ID to distinguish between the instances of choreography execution. The ID is generated upon a request to start a choreography instance. Discovery is necessary in any scheme where the potential service providers are unknown. Finally, event logging helps in debugging any error situations that eventually happen in distributed execution. Earlier, this approach has proven beneficial in message-broker-enabled choreographies [29]. Additionally, there can be local logging in the participating software, but the centralized logging records at least the messages being exchanged.

All execution support services help the UCH example as well. To start a choreography, the customer requests the framework to start an instance. The discovery helps especially when UCH brokering ask for price offers from the aggregators that are many and can enter or exit the business environment at any time. In these conditions, discovery removes the need of UCH brokering to explicitly know which aggregators to contact. Finally, the event logging service enables the monitoring and tracking of the choreography execution.

D. SERVICE COMMUNICATION LAYER

The execution of choreographies necessitates common concepts understood by all participants, hence a data model is necessary. Because Discografia focuses on functional interoperability and remains domain agnostic, it specifies no concrete data model but still acknowledges the need for such. In the electricity domain, whenever energy management is the concern, a potential candidate is Common Information Model (CIM) standardized as IEC 61970. The concrete syntax can be, e.g., XML or JavaScript Object Notation (JSON).

For the communication protocol, Discografia suggests the concept of message broker due to the application of the publish-subscribe paradigm. There are three reasons for this. First, the paradigm provides loose coupling in "time, space, and synchronization" [52]. Concretely, this means that for the participants to communicate, there is no need to know the exact network address or location of one another, and neither they need to strictly synchronize the processing of messages. Second, service choreographies inherently operate with asynchronous, event-driven communication [26, p. 192]. This is what message brokers have been designed for. Third, publish-subscribe enables topic-based one-to-many scenarios where one message is routed to multiple listeners, whoever is interested and authorized. The concrete examples of message brokers include but are not limited to the standards MQTT and Advanced Message Queueing Protocol (AMQP [53]) as well as the open-source product Apache Pulsar [54].

In the UCH brokering example, each participant must implement a support for the chosen data model and message broker. For instance, as the customer sends a request to the UCH broker, the data model enables a mutual understanding about the parameters of the request, such as the time slot of charging, the targeted state of charge, and the desired area. On the other hand, the message broker enables asynchronous communication and even topic-based routing to multiple participants without a need for manual configuration. The UCH broker needs to publish the request for price offers from aggregators only once, but the broker routes this to all the aggregators available, providing service discovery.

E. PARTICIPANT LAYER

1) EXPLICIT AND IMPLICIT EXECUTION

Discografia builds upon the idea of purely distributed choreographies, which means that choreography execution occurs in the software of the participants. The framework supplies the participants with a communication medium and the services necessary to enable the distributed execution.

The participants can execute a choreography *implicitly*, which means that any updates in the workflow are applied manually offline. This means that no workflow engine is necessary but the software developers will fetch the choreography specification and implement the required logic into the software considering the choreography role. The resulting application logic is, therefore, hard coded and cannot dynamically adapt to a new choreography version.

The other execution approach is *explicit*, where the choreography specification, i.e., the BPMN model, is executed with an interpreter engine. This is possible due to the executability of BPMN in any compliant process engine (such as Camunda [31] or SpiffWorkflow [32]). In principle, the explicit approach enables any workflow to be modeled by the choreography specifier and deployed to the participants automatically. However, BPMN cannot express all of the complexity required for execution, especially the details of data mapping and network communication. BPMN has no formal semantics either [55]. Furthermore, there are security implications in case of a malicious actor editing the specification.

The choice of explicit or implicit workflows is a trade-off between responsiveness to changes and the simplicity of software logic. The explicit approach is more flexible regarding changes. On the other hand, a change can be fully dynamic only when it is limited to the workflow expressed in BPMN. The implicit approach is most straightforward but requires additional manual work if the BPMN content of the choreography specification is complex. However, depending on the case, not all parts of the specification changes frequently and some choreography roles require only simple logic. Finally, regardless if the choice is explicit or implicit, the BPMN choreography specification always provides an agreement of the correct activities and the order of these. This contributes to functional interoperability compared to a situation either without a choreography specification or with one presented in a non-standard format.

The UCH brokering example provides potential for both implicit and explicit choreography execution. Implicit suits especially for the customer and DSO because their workflow contains only single task. In contrast, UCH brokering and aggregators have multiple steps to perform, which increases the advantages of explicit execution although implicit execution is still possible.

Regardless of whether explicit or implicit execution is in place, version control is essential for interoperability. That is, the participants must be able to recognize which version of the choreography to execute and potentially negotiate to find a version supported by all participants.

2) WORKFLOWS OF IMPLEMENTATION AND PARTICIPATION

To participate in choreographies via the framework, a choreography implementer must accomplish certain steps depending on whether it provides services or acts as a client to another participant. A service provider must make its offering available, whereas a participant must request for a choreography to start. Although the actual choreography-related communication occurs via the message broker, certain preliminary actions are necessary to implement the features of the framework, such as security, discovery, and choreography distribution.

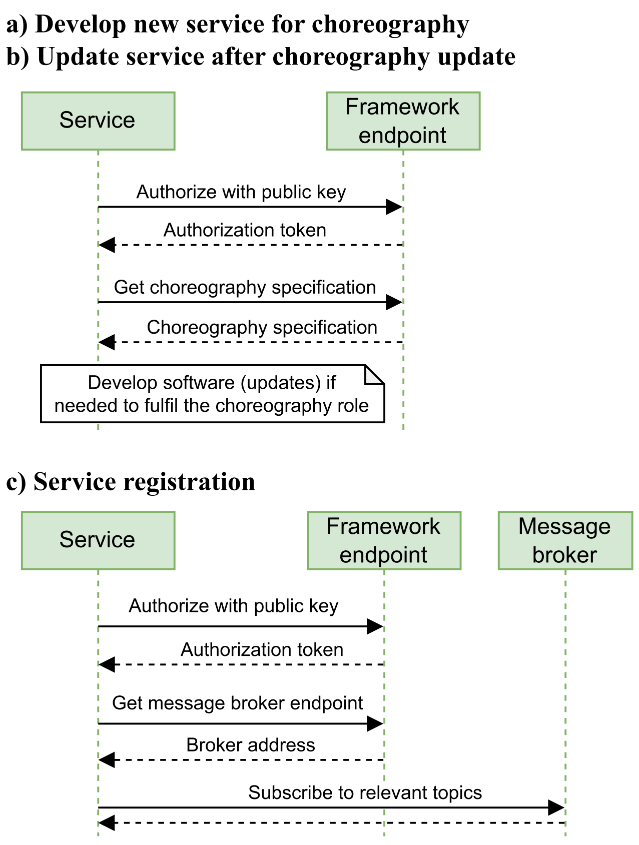
a) Develop new service for choreography
b) Update service after choreography update
Service
Framework endpoint
Authorize with public key
Authorization token
Get choreography specification
Choreography specification
Develop software (updates) if needed to fulfil the choreography role
c) Service registration
Service
Framework endpoint
Message broker
Authorize with public key
Authorization token
Get message broker endpoint
Broker address
Subscribe to relevant topics

FIGURE 5. The workflows of service setup.

Figure 5 illustrates the workflow of service development (a, b) and service registration (c). In all the workflows, the first step is authorization, where the participant provides its public key to the framework that, if the authorization succeeds, returns a token. Token-based authorization is a widely used state-of-art scheme and appears in, for instance, OAuth [56]. The workflow (a) depicts the development of a new service, whereas (b) refers to an update after a fresh choreography release. (a) and (b) are identical on a high level as both may require new software to be developed if the choreography specification necessitates functionality not yet supported. Once either (a) or (b) has finished, (c) enables participation by registering the service to the relevant topics in the message broker. The current (c) workflow omits any situations where the message broker or its address changes. Regarding (b), the workflow should be triggered by an update in the choreography specification. This can be triggered via a communication medium, such as a message broker, but the implementers can as well check for new choreography versions periodically.

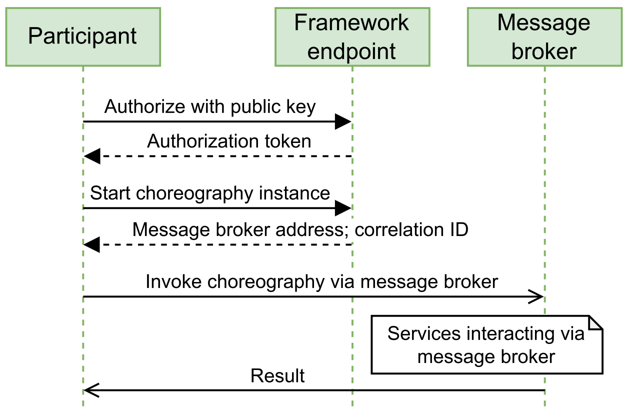
Participant
Framework endpoint
Message broker
Authorize with public key
Authorization token
Start choreography instance
Message broker address; correlation ID
Invoke choreography via message broker
Services interacting via message broker
Result

FIGURE 6. A possible workflow for choreography invocation.

Figure 6 depicts the invocation of a choreography. It assumes that the invocation is triggered by a customer expecting a business outcome although any event could invoke a choreography without any particular response message. Such an event could be, for example, the detection of an emergency where the subsequent activities can take long. In the illustrated workflow, the customer starts with authorization and then requests for the choreography to start. The start request generates a correlation ID to identify the choreography instance (enabling parallel execution) and the address of the message broker. The correlation ID will be included in the topic name to enable efficient message subscription and filtering by message broker clients.

Although the workflows appear complex compared to direct service invocations in the request-response fashion, it enables dynamic, adaptable choreographies. Any node can authorize itself, retrieve and implement the choreography specification, and discover other participants with the message-broker-based infrastructure (receiving the advantages of loose coupling due to publish-subscribe [52]). That is, dynamism and increased management possibilities come with the cost of additional complexity.

F. ADDITIONAL DESIGN CONSIDERATIONS

The design could still be extended to cover aspects that remain unspecified, the first of which is the management of the versions of the choreography specifications. There should be a support for metadata to enable references to certain specification versions (as the current BPMN specification [7] seems to omit such a feature). The participants would, then, negotiate about the version being executed for each choreography instance. For a concrete example, Transport Layer Security (TLS [57]), a mainstream technology for the encryption of data traffic, has exploited the multi-version approach for years, enabling a peer to select which versions to accept. Furthermore, the framework could provide a notification to implementers whenever a fresh version of a choreography is released. Any older versions could be declared obsolete due to security reasons or because of the price of maintaining a support for multiple versions.

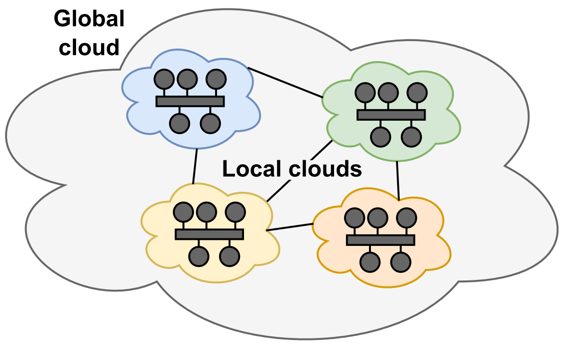

FIGURE 7. The global cloud of services consists of local clouds. Idea based on [63].

In the service communication layer, there is no tool to specify the data model and communication protocol because the choreographies are in BPMN that provides no such support. Additionally, there is no way to indicate which topic names to use in the message broker although these are necessary for interoperability. This question could be somehow included in the framework, or otherwise the participants might select a variety of non-interoperable technologies. The existing technologies for similar needs include at least Web Services Description Language (WSDL [58]) and OpenAPI [59] for service descriptions as well as XML Schema [60] and JSON Schema [61] for data models.

Furthermore, the framework is scalable only if it considers the interconnection of separate value chains or framework instances. The design builds all communication upon services, but there is currently no indication which organization would maintain the whole. Because Discografia is a framework rather than a platform, it assumes multiple instances. In a single logical entity, such as a city, there could be an agreement that all aggregators or UCH brokers connect to the same message broker and provide their services via the same framework instance, but this restricts any further scaling. On the other hand, Discografia is supposed to enable many kinds of services beyond UCH brokering and not restricting to the electricity domain. Then, even cross-ecosystem, cross-domain services would be possible and boost industrial innovations [62]. Therefore, for concrete interoperability regardless of the geographical location and service providers, there should be as many instances as needed and these should be interconnected. This vision, which is called local and global clouds and enables the consumption of services regardless of physical borders (see Figure 7), has earlier been introduced in Arrowhead framework [63]. The global cloud sets requirements especially to authorization to enable trust between the participants.

V. PROTOTYPE IN UCH BROKERING

To prove the choreography framework concept, this section presents an implementation for the UCH brokering process modeled in Section IV-A. Figure 8 depicts the system, showing which components are implemented in each layer. The participant layer has a component for each participant:

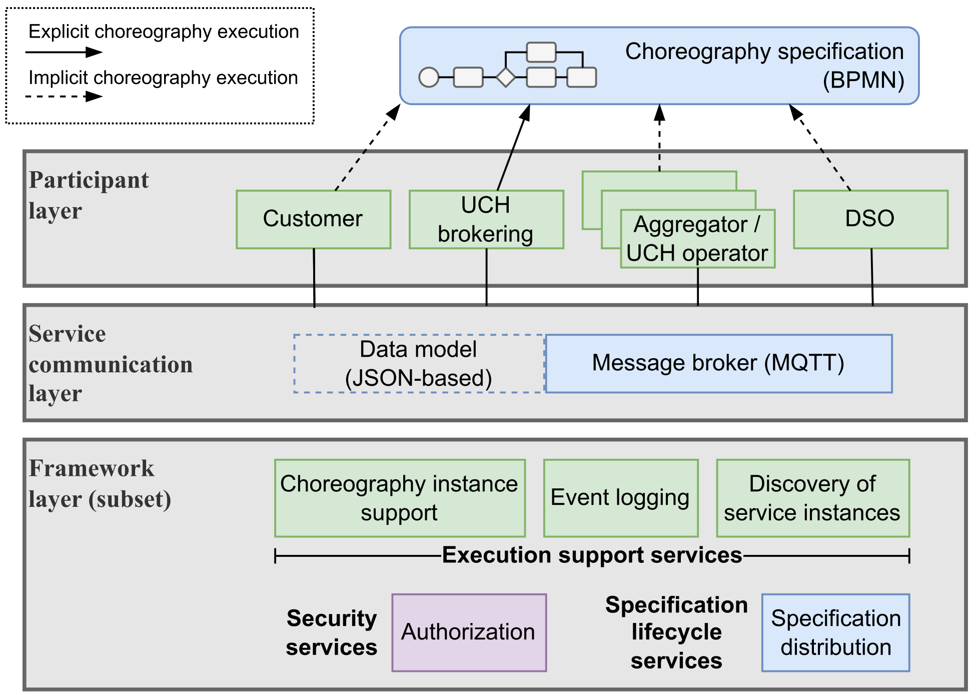
Explicit choreography execution
Implicit choreography execution
Choreography specification (BPMN)
Participant layer
Customer
UCH brokering
Aggregator / UCH operator
DSO
Service communication layer
Data model (JSON-based)
Message broker (MQTT)
Framework layer (subset)
Choreography instance support
Event logging
Discovery of service instances
Execution support services
Security services
Authorization
Specification lifecycle services
Specification distribution

FIGURE 8. The prototype system, including a partial implementation for the layers.

customer, UCH brokering service, aggregators, and DSO. The service communication layer contains MQTT as the message broker and a custom-made, JSON-based data model. The framework layer is a subset of the full framework, proving services for execution support, specification lifecycle, and security. The implementation of the layers is explained in more detail in the following subsections. Furthermore, the implementation is available as open source.[4]

Considering DSR method, this section provides a concrete, credible use case for the results for design and rigor cycles as introduced in Section IV. The use case and the related prototype prove the concept.

A. IMPLEMENTATION OF FRAMEWORK SERVICES

Table 2 shows how the services of the framework were developed, grouped after the categories security, lifecycle, execution, and service communication. The services are elaborated in the following paragraphs.

Among security and lifecycle services, which form the basis of choreography execution, authorization and specification distribution were developed. Both are REST endpoints. Authorization must occur before the framework can be used. The service is, however, a mockup that accepts any public key of the participant if provided as expected. It returns a token that the rest of framework services will accept. Specification distribution simply returns the current version of the BPMN process, which is physically an XML document.

For execution support services, there is choreography instance support, event logging, and the discovery of service instances. Instance support is a REST endpoint that generates a correlation ID, the unique identifier of a choreography instance. This ID is included in each MQTT topic name to enable the separation of messages. The event logging service is an MQTT client that subscribes for all choreography-related topics with a wildcard. It prints any received messages, enabling monitoring and debugging, but any larger-scale implementation would necessitates a logging database for more usability (similar to [29]). The discovery is based on MQTT topic names. That is, to discover any aggregators available, the UCH broker publishes its request to a topic that all aggregators listen to.

In the service communication layer, the message broker is HiveMQ [64], which is a commercial product but provides a

[4]https://gitlab.com/tau-tase/discografia

TABLE 2. The implementation of the framework services in the prototype.

Service	Implementation	Platform
Security services		
Authorization	HTTP REST endpoint (mockup)	Python, Docker
Specification lifecycle services		
Specification distribution	HTTP REST endpoint	Python, Docker
Execution support services		
Choreography instance support	HTTP REST endpoint	Python, Docker
Event logging	MQTT client	Python, Docker
Discovery of service instances	(Based on MQTT topics)	–
Service communication layer		
Message broker	HiveMQ	Docker

TABLE 3. The implementation of the participants in the prototype.

Participant – Enclosed component	Choreography execution	Implementation[1]
Customer	Implicit	Python application
UCH broker	Explicit	Java application
– Camunda BPMN parser		Java component
– Smart Routing algorithm[2]		Python web service
Aggregator	Implicit	Python application
– Dynamic Pricing algorithm[2]		Python web service
DSO	Implicit	Python application

[1]All participants and enclosed components have a dedicated Docker container.

[2]Introduced in [47].

free version with limited functionality. HiveMQ is extensible and therefore a candidate for future implementations with the Discografia-specific authorization that no existing product can provide. The concept of customized authorization schemes aligns with the philosophy of MQTT standard, which states that "it is the implementer's responsibility to include the appropriate features as part of their design" [46].

B. IMPLEMENTATION OF PARTICIPANTS

Table 3 provides an overview of the details of participating software. All of these operate as a Docker container. Only the UCH brokering service executes the choreography specification (see Section IV-A) explicitly, whereas the others rely on an implicit, hard-coded implementation. The implicit version can be faster to develop but lacks the advantages of adaptability if the choreography specification is updated. However, both the customer and DSO have a little of choreographical interaction, as the customer performs solely a single request and DSO receives only one message per choreography instance. All of the functionality, as explained in the following paragraphs, is based on the choreography specification in Figure 3.

The customer performs a single request to receive a UCH and aggregator to go to, but certain preparations are necessary for the choreography. First, the customer must authorize itself with its public key. Next, it retrieves the latest version of the choreography specification in case of modifications. Finally, it performs another call to the framework to instantiate the choreography and to receive the information required, namely the related correlation ID and the address of the message broker. The correlation ID is included in each choreography-specific topic name and therefore enables a separation of messages per the choreography instance. Finally, the customer publishes a request to the UCH brokering service via MQTT and waits for a reply.

Once the request has appeared from the customer, the UCH brokering service asks for price and availability information from the aggregators and decides which to choose for the customer for EV charging. This choice is based on the algorithm called Smart Routing, which the UCH broker executes. This algorithm was earlier introduced in [47]. The algorithm has been implemented as a mixed integer linear program utilizing the optimization modeling library, Pyomo [65]. The specific instances of the Smart Routing algorithm are solved with the open-source optimization solver GLPK [66], which has been containerized within the computer program run in UCH brokering.

Because UCH brokering executes the choreography explicitly, it needs a BPMN parser, for which Camunda [31] was chosen. Camunda (made in Java) was chosen due to its comprehensive documentation. In addition to mere parsing, Camunda provides a complete BPMN engine, but it remains future work to integrate such an engine into the prototype.

The task of the aggregators is to wait for requests from UCH brokering, then provide price information and, if the UCH broker selects the aggregator, send the updated net behavior to the DSO. The generation of prices re-uses Dynamic Pricing algorithm published in [47]. The Python implementation of the algorithm can be found in datafev framework [67].

In the choreography, the DSO receives only a message about the net behavior of the selected aggregator. There is no need for any awareness about the choreography instance, because the reporting of net behavior is a generic operation. Still, the DSO must be authorized and know the address of the message broker similar to all other nodes.

Listing 1 shows the MQTT messages logged during a test run. Each line starts with a timestamp, followed by the MQTT topic name and the actual message. In each topic name, the substring starting with "corr-" is the correlation ID. The messaging sequence starts with the customer publishing a request to the UCH brokering service, after which this asks for price information from the aggregators. Then, the UCH broker selects one of the aggregators, rejecting the others. The full messaging sequence took approximately 3 seconds, which is a reasonable response time considering everything executed as Docker containers in a virtual machine in a laptop. Still, the full execution time varied between 40–60 seconds, but most of it was consumed waiting for the startup of the containers. In a real implementation with proper resources for each node, the execution should take a few seconds at maximum to enable a decent user experience. The development environment was Ubuntu 20.04 in a virtual machine (VM) operated in a laptop, and the VM had 6 GB of RAM and three processor cores.

The prototype shows that the choreography framework concept can operate in practice. Although there is a centrally managed choreography specification, the participants executed their part on their own, which resulted in a fully distributed execution. The participants were authorized to enable trust. To enable the communication, the framework provided the address of the message broker and fulfilled the discovery of aggregators with MQTT messaging. Finally, the framework logged all execution, enabling monitoring and debugging if any incompatibility would occur between the participants.

```
$ docker logs discografia-framework-monitoring-1
Subscribed with topic pattern: 'discografia/#'
10:34:12.018252 discografia/corr-1685097251.8024755/UchBroker/ReservationRequest/uchbroker1: {
  "customer_id": "customer_x",
  "battery_capacity_kwh": 110,
  "v2g_allowance_kwh": 0,
  "soc_arrival": 0.4,
  "soc_target": 0.8,
  "power_charge_kw": 11,
  "power_discharge_kw": 11,
  "time_arrival": "2023-05-26T10:44:00+00:00",
  "time_departure": "2023-05-26T12:14:00+00:00"
}
10:34:12.214070 discografia/corr-1685097251.8024755/Aggregator/OfferRequest: {
  ...
}
10:34:12.305094 discografia/corr-1685097251.8024755/Aggregator/OfferResponse/AggregatorC: {
  "charger_id": "cu_AggregatorC_03", ...
}
10:34:12.326026 discografia/corr-1685097251.8024755/Aggregator/OfferResponse/AggregatorF: {
  "charger_id": "cu_AggregatorF_03", ...
}
10:34:12.373248 discografia/corr-1685097251.8024755/Aggregator/OfferResponse/AggregatorN: {
  "charger_id": "cu_AggregatorN_03", ...
}
10:34:14.438119 discografia/corr-1685097251.8024755/Aggregator/ReservationAcceptRequest/AggregatorN: {
  "charging_unit_id": "cu_AggregatorN_03",
  "response": "accept"
}
10:34:14.822841 discografia/corr-1685097251.8024755/Aggregator/ReservationAcceptResponse/AggregatorN: {
  "response": "ok"
}
10:34:14.822951 discografia/Dso/NetBehavior: {
  "aggregator_id": "AggregatorN",
  "net_behavior": "..."
}
10:34:14.829505 discografia/corr-1685097251.8024755/Aggregator/ReservationRejectRequest/AggregatorC: {
  "response": "reject"
}
10:34:14.831774 discografia/corr-1685097251.8024755/Aggregator/ReservationRejectRequest/AggregatorF: {
  "response": "reject"
}
10:34:14.833364 discografia/corr-1685097251.8024755/UchBroker/ReservationResponse/uchbroker1: {
  "charging_unit_id": "cu_AggregatorN_03"
}
```

Listing. 1. Log output from the monitoring service, showing the timestamps and topic names of the MQTT messages exchanged between the participants (some JSON content replace with "…" to save space). The aggregator "N" was selected for charging.

VI. WORK AMOUNT EVALUATION

Although Section V provides a proof of concept, this section further shows the expected advantages of the proposed software system framework. Considering DSR method, this section is a quantitative indication of the benefits resulting from the design cycle and contributing to the scientific rigor.

Discografia can be evaluated quantitatively with the hypothesis that it reduces the amount of work required in the implementation and maintenance of choreographies. The hypothesis receives support from [68, p. 14]. Still, the concrete advantages depend on the degree of automation in communication and the software development required for implementation.

The advantages can be quantified by estimating the overall manual work required in choreographing. Work is necessary whenever a new choreography is created or an existing one is modified. Let the total work be W, defined as the sum of individual work items: W_{spe} for specifying the choreography, W_{com} for communicating it to any interested parties, and W_{dev} for developing the required changes in whichever system or software that implements a part of the choreography. This is expressed as in (1).

$$W = W_{spe} + W_{com} + W_{dev} \tag{1}$$

The extent of some but not all work items depends on the number of parties. Let the number of implementers in the business network be n. It can be assumed that W_{spe} occurs only once for each choreography version regardless if n is 2, 10, or 1,000. However, W_{com} and W_{dev} must occur for each choreography implementer (see Figure 9). Thus, $W(n)$ becomes (2).

$$W(n) = W_{spe} + \sum_{i=1}^{n} (W_{com}(i) + W_{dev}(i)) \tag{2}$$

In the UCH brokering example, the work items would appear as follows during an update. W_{spe} is the effort required to model the choreography, similar to Figure 3. W_{com} is necessary to communicate the specification to all implementers, that is, at least UCH brokers and aggregators. Respectively, W_{dev} is required from UCH brokers and aggregators to develop any software updates required.

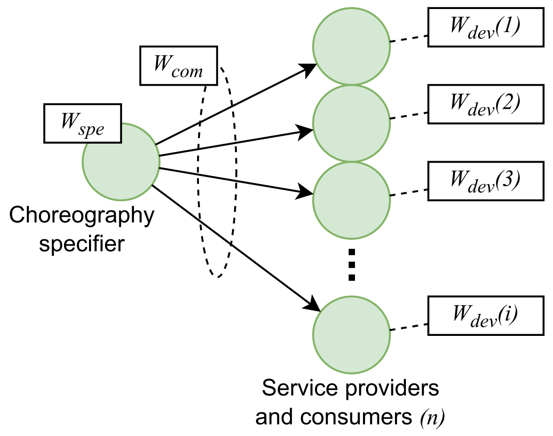

FIGURE 9. The work items required in choreography releases.

For simplicity, let us assume that W_{com} and W_{dev} are equal in all parties. Although this is untrue in a single choreography release, the overall W_{com} and W_{dev} are proportional to n in the large scale over a longer period with repeated choreography releases. $W(n)$ becomes (3).

$$W(n) = W_{spe} + n(W_{com} + W_{dev}) \tag{3}$$

Let us consider four scenarios, S_0–S_3, with a varying amount of automation and therefore manual work (see Table 4). W_{spe} is assumed always manual because it requires a human user to specify the business process explicitly. Let the work amount of W_{spe} be x. In S_0, both W_{com} and W_{dev} are fully manual too. For simplicity, let both of these require x amount of work as well. In S_1, the business network takes a partial advantage from ICT tools in choreographing, reducing W_{com} and W_{dev} by 50 %, which drops the amount of work to $0.5x$ for both individually. For W_{com}, such an advantage can occur when the choreography is distributed in an electronic format but still requires manual implementation in the receiving system. Respectively, W_{dev} reduces when the execution logic is generated automatically but still requires manual programming to be complete. In S_2, W_{dev} remains at $0.5x$, but W_{com} is fully automatic and therefore practically 0. Finally, S_3 assumes full automation in both W_{com} and W_{dev}. It is notable that W_{dev} is the most difficult item to automatize. Whether this is feasible depends not only on technology but also the simplicity of the new choreography logic. It should be possible to describe all of the logic in the respective choreography specification.

Therefore, W depends on not only n but also s, the scenario. We get (4).

$$W(n, s) = W_{spe}(s) + n(W_{com}(s) + W_{dev}(s)) \tag{4}$$

As we assign values from Table 4 to (4), we get, for example, (5) and (6).

$$W(10, S_0) = x + 10(x + x) = 21x \tag{5}$$

$$W(10, S_1) = x + 10(0.5x + 0.5x) = 11x \tag{6}$$

By calculating W with (4), we get Table 5, which groups the work amount calculated for each scenario s by the implementer count n being either 10, 100, or 1000. In the rightmost column, the table shows the relative $W(n, s)$ as compared to $W(n, S_0)$, which assumes automation in neither W_{com} nor W_{dev} (i.e., choreography-related distribution and software updates). For example, $W(10, S_1)$ is only 52.4 % of $W(10, S_1)$ and $W(10, S_2)$ is 28.6 % respectively. The calculations show that the larger the network, the more advantage the choreography framework can provide. Yet the benefits depend on the degree of automation in W_{com} and W_{dev}.

Similarly in the UCH brokering example, the overall advantages are higher when the size of network grows. The current example contains only one UCH broker, but these could be many as well. On the other hand, the number of recharging service providers could be higher. The aggregators could be joined by microgrids [12] or smaller-scale prosumers [11] that produce, for instance, solar electricity. Such a business scheme would lead to a growing number of participants and therefore receive a considerable advantage from automatically distributed and executed choreographies.

TABLE 4. Scenarios and the amount of manual work they require for a choreography release.

s – scenario	W_{spe}	W_{com}	W_{dev}
S_0 – no automation	x	x	x
S_1 – partial automation	x	$0.5x$	$0.5x$
S_2 – partial automation	x	0	$0.5x$
S_3 – full automation	x	0	0

W_{spe}: Work for specification
W_{com}: Work for communication to implementers
W_{dev}: Work for software development by implementers

TABLE 5. The relative benefit regarding manual work in each scenario when n grows.

n	s	$W(n,s)$	$W(n,s)/W(n,S_0)$
10	S_0	$21x$	100.0 %
	S_1	$11x$	52.4 %
	S_2	$6x$	28.6 %
	S_3	x	4.8 %
100	S_0	$201x$	100.0 %
	S_1	$101x$	50.2 %
	S_2	$51x$	25.4 %
	S_3	x	0.5 %
1000	S_0	$2001x$	100.0 %
	S_1	$1001x$	50.0 %
	S_2	$501x$	25.0 %
	S_3	x	< 0.1 %

n: Number of choreography implementers
$W(n, s)$: Manual work total

VII. ASSESSMENT OF RESULTS

The main contribution of this study is to suggest an architectural framework as a solution to problems that have not yet been addressed in power systems, in particular from the viewpoint of value chains and business networks. Currently, there is no off-the-shelf solution for the problem although there are software tools that can serve as the foundational building blocks. To elaborate the contribution of this work, this section provides an overview and critical analysis of the suggested software-architectural framework.

From the DSR viewpoint, this section elaborates the results. It does not provide new results to the three cycles but rather explains the expected value and therefore contributes to the communication of the scientific rigor.

A. NOVELTY, ORIGINALITY, AND SIGNIFICANCE

The novelty of this work comes from the proposed method, that is, the software system framework for the distributed execution and the communication of choreography specifications, built upon standard-based technologies. The selected standards are BPMN, MQTT, and Restful API technologies, but they could be any respective standards. Although software systems communicate even today, the current approaches lack a similar level of scalability regarding manual work in the efforts required for interoperability (see Section VI). It can be claimed that the limitations of current software systems hinder new added value and services from appearing, because there is too much manual work in the communication of choreography specifications and no system to help in distributed choreography execution.

Respectively, the originality comes from the new understanding regarding the framework design. The design (Section IV) suggests a framework and its functionality, the prototype (Section V) is an indication of feasibility, and the evaluation (Section VI) suggests quantitative value. No earlier work has provided similar design knowledge for the distributed execution of choreographies.

Based on Section III-F, this research is significant as no earlier work has reached the requirements of distributed choreography execution and the management of choreography specifications. The requirements cover distributed, inter-organizational choreography execution, common modeling language, supporting services, lightweight system-to-system communication, and application in power systems. This significance applies generally across the application domains of ICT. Yet, service choreographing, as a tool for functional interoperability, appears completely unexplored in the power system domain (see Section III-A). Thus, the results bring a valuable contribution to the overall servitization of the power system.

B. THEORETICAL AND PRACTICAL CONTRIBUTIONS

Theoretically, this work contributes to interoperability and service-oriented systems that operate across value chains. As illustrated in Figure 2, the interoperability model EIF has four layers: technical, semantic, organizational, and legal, and organizational interoperability can be seen to enclose functional interoperability. As argued in Section III, earlier works have not provided a similar contribution to functional interoperability across organizational borders based on a service framework to communicate choreography specifications and to support the actual execution. On the other hand, current service architectures either focus on the technical and semantic levels or operate in a value chain restricted to homogeneous components, often from a single vendor or platform (such as the provider of industrial control systems or cloud services).

Furthermore, the practical contributions of the article are expected to show in the form of facilitated value chain coordination in the future. The choreography framework remains abstract and has this far no prototype with hardware equipment included. Once the concept is implemented in a physical system, the advantages suggested in Section VI are expected to materialize regarding the amount of manual work saved. Because Discografia is a domain-agnostic framework, the use case can, in principle, be anything that encloses cross-organizational, distributedly executed service choreographies.

Considering UCH brokering, this work brings the use case closer to an actual service-oriented environment and helps in functional interoperability. It introduces service connections between the participants and enables a computationally executable choreography that forms an agreement between the participants. The agreement can be updated, and the participants can agree to support the new version. This enables functional interoperability that stands the evolution in the operation of the value chain. Respective advantages can be expected in the other collaborative multi-actor business processes of the power system now that the conventional actor hierarchy is transforming into a more flexible service-based scheme (see Section III-A).

C. SOFTWARE STABILITY

Discografia is a software system and can therefore be assessed in terms of software stability, a concept concerned with persistent reusability. Software stability affects the amount of re-engineering that is necessary to meet the required evolution during software lifecycle and is therefore an essential factor regarding the feasibility of any software product [69]. The concept stems from object-oriented programming but is applicable in systems of systems, including Discografia, due to its abstract nature. In software design, a way to increase stability is to model objects after their function, or the business need, rather than after the concrete appearance of the objects [70]. Stability can be analysed from abstract software models and from multiple viewpoints. For example, AbuHassan and Alshayeb [71] propose a metrics for structural, functional, and behavioral stability.

From the start, the design of Discografia has aimed towards re-usable structures that are abstract and therefore reusable across business problems and domains. Still, it can be argued that some parts are stabler than others although there is no way to predict the future.

The layered architecture (see Figure 4) provides a solid basis for stability. All of the layers (participant, service communication, and framework) focus purely in high-level functionality, creating layers of abstraction between one another. These layers are unlikely to change.

The layers are further split into elements (see Figure 4) that are assumed to remain useful. Although these are finer grained, the elements have a function, such as execution, data model, or discovery, clearly separated from the implementation. The most concrete of these elements is *message broker*, which includes the idea of a certain type of messaging medium usually based on the publish-subscribe approach. Still, publish-subscribe is a generic basic pattern and unlikely to completely vanish from networked systems. Even if there were a need to re-engineer for another, more powerful communication pattern, this would likely be complementary design.

For additional functionality, there have already been examples of complementing an existing framework. For instance, Arrowhead framework has received additional services, such as the workflow choreographer [37] after the initial publication. Yet the earlier services of Arrowhead, such as authorization, service registry, and orchestration, remain in operation.

D. SUITABILITY

There are at least four arguments that support the suitability of the proposed communication framework. First, it builds upon commonly used service concepts, such as authorization, discovery, common data models, and microservices. Discografia is a continuation from these concepts that already deliver results in everyday systems. Second, the service framework Arrowhead [63] has earlier provided a generic, domain-agnostic concept that has resulted in years of additional research and extensions. Similar advantages can be expected to apply to Discografia that expands the service concept to distributed choreography execution. Third, the software prototype in Section V provides a proof of concept to show how the various components retrieve the choreography specification over Internet and then proceed to execute the choreography together. Fourth, the quantitative evaluation in Section VI suggests general advantages from the concept regardless of the use case. The advantages grow when the size of network participants grows, suggesting scalability, which is often problematic when the size of a distributed system grows.

As shown in Section V, the prototype system suggests suitability for the UCH brokering use case. Discografia enabled the UCH brokering process to execute and to find the most appropriate charging spot for the EV depending on the load and availability conditions of the charging services. Besides, Discografia provided the background functionality provided the essential auxiliary services and features, such as interoperability, authorization, the distribution of choreography specifications, and service discovery. Therefore, Discografia appears to support the business model where UCH brokering and charging services are distributed and even the load conditions of the distribution system are considered. This is aligned with the goals of the use case. A similar suitability can be expected in power systems in general whenever the value chain includes multiple actors or organizations that must perform a predefined set of activities to reach a common business goal.

Because the majority of the work has occurred in a conceptual level, the concrete future solutions can differ regarding openness and the chosen technologies. A limitation is that the proof of concept considers only a single use case and remains conceptual, not including physical actors or equipment. Still, because the included design concepts are generic as suggested in Subsection VII-C, it is safe to assume that the framework design is transferable across use cases. However, when software technologies enter the market, there are always business-related reasons that affect the final outcome. Often, the highest commercial power wins, which may give an advantage to proprietary solutions instead of standard-based, open designs. This does not reduce the conceptual value of Discografia but reminds that the idealisms of free software and open standards do not always win.

VIII. DISCUSSION

This article carries out design science research (DSR) to introduce a novel service choreography framework for the electric power system of the future. The research method includes three cycles: rigor for scientific contribution, design for the research work itself, and relevance for everyday benefits [10]. The rigor cycle closes with the scientific contribution from the choreography framework concept, whereas the design cycle was an internal effort during the research. The third cycle, relevance, will become fully concrete only in the future as explicitly specified service choreographies enter everyday systems. In DSR, the designed artifacts are essential, forming the concrete outcome. The most significant artifact is the framework concept, whereas the prototype is a secondary artifact proving the concept.

The results suggest that the choreography framework is a suitable approach for functional interoperability in a distributed, inter-organizational scheme. The use case is UCH brokering, that is, the selection of the chargers available for the EVs based on the desired time slot, location for charging, and aggregator pricing, considering the availability and the scheduled load of aggregators. In the prototype, a choreography is executed, explicitly by one actor and implicitly by the others.

The research work has limitations regarding the coverage of the software implementation. Of the parts of the framework, the proof of concept implemented execution support services, specification distribution, and a mock-up for authorization. That is, the framework layer received only a partial implementation, and the functionality of the remaining parts could be developed in another prototype. These parts include a proper user authentication and authorization as well as the rest of specification lifecycle services. On top of security, the achievement of trust between the participants could be researched further.

Currently, the work considers only one example process, namely UCH brokering. More processes would provide more coverage and either provide evidence that the framework is widely applicable or reveal shortcomings.

On the other hand, the prototype is executed only inside one computer without any actual hardware or system interfaces connected. Although UCH brokers are still a future concept, the prototype could at least connect with actual DSO interfaces and EVs.

The framework could provide more support for monitoring choreography execution. There could be status tracking along with an API for graphical user interfaces to monitor the status of processes, including potential errors. On the other hand, the framework could provide functionality for error handling and especially notifications to choreography participants in case the execution appears to fail. This would enable error recovery in the participating software. Still, such centralized features should be designed sparingly to keep the concept distributed and therefore scalable.

The framework concept is evaluated quantitatively with the work amount evaluation. This calculates the expected reduction of the manual effort required for the communication and implementation of choreographies. The calculation assumes that choreography implementations are less laborious thanks to the automation reached with the visual, computationally interpretable modeling technique of BPMN. However, because this evaluation is theoretical, there could be more concrete evidence.

IX. CLOSURE

A. CONCLUSIONS

This work suggests the software framework Discografia to enable distributed service choreographing with supporting services for modeling and execution. The framework comprises three main layers. First, *framework layer* provides generic services for security, specification lifecycle management, and execution support. Second, *service communication layer* defines the common communication protocols and data models to enable information exchange. Third, *participant layer* includes the actual choreography participants that provide services and/or consume these.

In this work, the service choreography refers to a collaborative workflow between participants. Choreographies can be explicitly specified (or modeled) to define the interaction as a sequence of activities, events, and conditions. Such specifications enable functional interoperability. In Discografia, the modeling notation is BPMN 2.0, which is not only visual but also computationally executable.

To actually execute choreographies, the participants receive the related specification from Discografia. Then, the participants accomplish their part of the specification. Thus, the execution is distributed but aims at a common goal. During execution, the participants interact via the service communication layer.

This work evaluates the framework concept in two ways: with a software prototype demonstration and with a quantitative calculation of the expected reduction in choreographing-related efforts compared to existing manual methods. First, the prototype proves the concept, showing that independent actors can execute collaborative workflows in a distributed way with the help of the framework. On the other hand, the concept enables heterogeneous software platforms, as Python and Java co-exist in the prototype. Second, the quantitative evaluation suggests that the wider the value network, the more overall advantage is reached. The advantage comes from reducing the effort required for the communication and implementation of the choreography specifications.

Discografia brings novelty regarding at least two main factors. First, the execution of choreography models occurs in a fully distributed way as the framework provides only what is necessary to manage the specifications and to support the communication of participants. Second, the modeling and execution of service choreographies seem previously unexplored in the power systems domain.

B. FUTURE WORK

There are multiple possibilities for future research. As outlined in Section IV-F, the further design of the framework could consider the version management and negotiation of choreography specifications, the specification of data models and the communication protocol, and the interconnection of local and global service clouds.

The prototype could implement all of the framework features, such as a proper authorization method, and specification lifecycle services. On the other hand, a standard-based data model could be chosen, potentially IEC 61970. Furthermore, a participant could integrate a full workflow engine instead of a plain BPMN parser. As a longer-term goal, the prototype could consider data autonomy and sovereignty to guarantee trust similar to International Data Spaces (IDS [50]) and Gaia-X [51].

On the other hand, distributed-ledger-based systems have been proposed for distributed service choreographies [38], [39], [40], [42], [43]. In Discografia, the network builds the trust upon key- and token-based authorization. The approach is simpler compared to distributed ledgers and therefore considered more suitable for IoT, electric mobility, and other mobile machines. Nevertheless, to resolve the concrete implications of this difference, there could be a comparative work.

Finally, more use cases could be explored. Service-oriented architectures are exploited in multi-party schemes in a variety of domains, such as industrial production [28], smart cities [72], and mobile machinery [49]. All of these areas could benefit from distributed choreography modeling and execution especially when the number of choreography implementers is high and they come from separate organizations.

REFERENCES

[1] C. Keating, R. Rogers, R. Unal, D. Dryer, A. Sousa-Poza, R. Safford, W. Peterson, and G. Rabadi, "System of systems engineering," *Eng. Manag. J.*, vol. 15, no. 3, pp. 36–45, 2003.

[2] Y. Parag and B. K. Sovacool, "Electricity market design for the prosumer era," *Nature Energy*, vol. 1, no. 4, p. 16032, Mar. 2016.

[3] A. Kulmala, M. Baranauskas, A. Safdarian, J. Valta, P. Järventausta, and T. Björkqvist, "Comparing value sharing methods for different types of energy communities," in *Proc. IEEE PES Innov. Smart Grid Technol. Eur. (ISGT Europe)*, Oct. 2021, pp. 1–6.

[4] M. Gržanic, T. Capuder, N. Zhang, and W. Huang, "Prosumers as active market participants: A systematic review of evolution of opportunities, models and challenges," *Renew. Sustain. Energy Rev.*, vol. 154, Feb. 2022, Art. no. 111859.

[5] *A Compilation of IEEE Standard Computer Glossaries*, IEEE Standard 610, 1991, doi: 10.1109/IEEESTD.1991.106963.

[6] European Commission. (2017). *New European Interoperability Framework*. Accessed: Nov. 29, 2022, doi: 10.2799/78681.

[7] (2013). *Business Process Model and Notation (BPMN) Version 2.0.2*. OMG. [Online]. Available: http://www.omg.org/spec/BPMN

[8] S. Baskarada, V. Nguyen, and A. Koronios, "Architecting microservices: Practical opportunities and challenges," *J. Comput. Inf. Syst.*, vol. 60, no. 5, pp. 428–436, Sep. 2020.

[9] R. Baskerville, A. Baiyere, S. Gergor, A. Hevner, and M. Rossi, "Design science research contributions: Finding a balance between artifact and theory," *J. Assoc. Inf. Syst.*, vol. 19, no. 5, pp. 358–376, May 2018.

[10] A. R. Hevner, "A three cycle view of design science research," *Scandin. J. Inf. Syst.*, vol. 19, no. 2, pp. 87–92, 2007.

[11] R. Zafar, A. Mahmood, S. Razzaq, W. Ali, U. Naeem, and K. Shehzad, "Prosumer based energy management and sharing in smart grid," *Renew. Sustain. Energy Rev.*, vol. 82, pp. 1675–1684, Feb. 2018.

[12] I. Zenginis, J. S. Vardakas, C. Echave, M. Morató, J. Abadal, and C. V. Verikoukis, "Cooperation in microgrids through power exchange: An optimal sizing and operation approach," *Appl. Energy*, vol. 203, pp. 972–981, Oct. 2017.

[13] M. Tanjimuddin, P. Kannisto, P. Jafary, M. Filppula, S. Repo, and D. Hästbacka, "A comparative study on multi-agent and service-oriented microgrid automation systems from energy internet perspective," *Sustain. Energy, Grids Netw.*, vol. 32, Dec. 2022, Art. no. 100856.

[14] S. Borenius, P. Kekolahti, H. Hämmäinen, M. Lehtonen, and P. Mähönen, "Novel industry architectures for connectivity solutions in the smart distribution grids," *IEEE Access*, vol. 11, pp. 68093–68112, 2023.

[15] M. Pau, E. Patti, L. Barbierato, A. Estebsari, E. Pons, F. Ponci, and A. Monti, "Design and accuracy analysis of multilevel state estimation based on smart metering infrastructure," *IEEE Trans. Instrum. Meas.*, vol. 68, no. 11, pp. 4300–4312, Nov. 2019.

[16] (May 8, 2023). *SOGNO LF Energy*. [Online]. Available: https://lfenergy.org/projects/sogno/

[17] M. Pau, M. Mirz, J. Dinkelbach, P. Mckeever, F. Ponci, and A. Monti, "A service oriented architecture for the digitalization and automation of distribution grids," *IEEE Access*, vol. 10, pp. 37050–37063, 2022.

[18] I. Losa, A. Monti, M. Ginocchi, V. Croce, F. Bosco, E. de Luca, G. Fedele, D. Stratogiannis, and B. Petters, "PLATONE: Towards a new open DSO platform for digital smart grid services and operation," in *Proc. 26th Int. Conf. Exhib. Electr. Distribution*, Sep. 2021, pp. 2974–2978.

[19] M. Haghgoo, A. Dognini, T. Storek, R. Plamanescu, U. Rahe, S. Gheorghe, M. Albu, A. Monti, and D. Müller, "A cloud-based service-oriented architecture to unlock smart energy services," *Energy Informat.*, vol. 4, no. 1, p. 9, Dec. 2021.

[20] N. Kabbara, M. O. Nait Belaid, M. Gibescu, L. R. Camargo, J. Cantenot, T. Coste, V. Audebert, and H. Morais, "Towards software-defined protection, automation, and control in power systems: Concepts, state of the art, and future challenges," *Energies*, vol. 15, no. 24, p. 9362, Dec. 2022.

[21] G. Paludetto, E. Bionda, and F. Soldan, "MESP—An interoperable platform for multi-energy systems," in *Proc. AEIT Int. Annu. Conf. (AEIT)*, Oct. 2022, pp. 1–6.

[22] (2012). *Smart Grid Reference Architecture 3.0*. CEN-CENELEC-ETSI Smart Grid Coordination Group. Accessed: Nov. 29, 2022. [Online]. Available: https://www.cencenelec.eu/media/CEN-CENELEC/AreasOfWork/CEN-CENELEC_Topics/SmartGridsandMeters/SmartGrids/reference_architecture_smartgrids.pdf

[23] C. Peltz, "Web services orchestration and choreography," *Computer*, vol. 36, no. 10, pp. 46–52, Oct. 2003.

[24] A. Dorri, S. S. Kanhere, and R. Jurdak, "Multi-agent systems: A survey," *IEEE Access*, vol. 6, pp. 28573–28593, 2018.

[25] M. Dalla Preda, M. Gabbrielli, S. Giallorenzo, I. Lanese, and J. Mauro, "Dynamic choreographies: Theory and implementation," *Log. Methods Comput. Sci.*, vol. 13, no. 2, pp. 1–57, 2017.

[26] C. Surianarayanan, G. Ganapathy, and R. Pethuru, *Essentials of Microservices Architecture: Paradigms, Applications, and Techniques*. Boca Raton, FL, USA: Taylor & Francis, 2019.

[27] U. Bharti, A. Goel, and S. C. Gupta, "ReactiveFnJ: A choreographed model for fork-join workflow in serverless computing," *J. Cloud Comput.*, vol. 12, no. 1, p. 63, Apr. 2023.

[28] P. Kannisto, D. Hästbacka, T. Gutiérrez, O. Suominen, M. Vilkko, and P. Craamer, "Plant-wide interoperability and decoupled, data-driven process control with message bus communication," *J. Ind. Inf. Integr.*, vol. 26, Mar. 2022, Art. no. 100253.

[29] P. Kannisto, V. Heikkilä, O. Hylli, M. Attar, S. Repo, and K. Systä, "SimCES platform for modular simulation: Featuring platform independence, container ecosystem, and development toolkit," *SoftwareX*, vol. 19, Jul. 2022, Art. no. 101189.

[30] G. Decker and M. Weske, "Interaction-centric modeling of process choreographies," *Inf. Syst.*, vol. 36, no. 2, pp. 292–312, Apr. 2011.

[31] *Camunda*. Accessed: Apr. 26, 2023. [Online]. Available: https://camunda.com

[32] *SpiffWorkflow*. Accessed: Apr. 26, 2023. [Online]. Available: https://www.spiffworkflow.org

[33] *WSO$_2$ Enterprise Integrator*. Accessed: May 12, 2023. [Online]. Available: https://ei.docs.wso2.com/en/latest/

[34] A. Ottensooser, A. Fekete, H. A. Reijers, J. Mendling, and C. Menictas, "Making sense of business process descriptions: An experimental comparison of graphical and textual notations," *J. Syst. Softw.*, vol. 85, no. 3, pp. 596–606, Mar. 2012.

[35] M. Cortes-Cornax, S. Dupuy-Chessa, D. Rieu, and N. Mandran, "Evaluating the appropriateness of the BPMN 2.0 standard for modeling service choreographies: Using an extended quality framework," *Softw. Syst. Model.*, vol. 15, no. 1, pp. 219–255, Feb. 2016.

[36] L. D. Nguyen, A. Bröring, M. Pizzol, and P. Popovski, "Analysis of distributed ledger technologies for industrial manufacturing," *Sci. Rep.*, vol. 12, no. 1, p. 18055, Oct. 2022.

[37] D. Kozma, P. Varga, and F. Larrinaga, "Dynamic multilevel workflow management concept for industrial IoT systems," *IEEE Trans. Autom. Sci. Eng.*, vol. 18, no. 3, pp. 1354–1366, Jul. 2021.

[38] L. Spalazzi, F. Spegni, A. Corneli, and B. Naticchia, "Blockchain based choreographies: The construction industry case study," *Concurrency Comput., Pract. Exper.*, vol. 35, no. 16, p. e6740, Jul. 2023.

[39] O. López-Pintado, L. García-Bañuelos, M. Dumas, I. Weber, and A. Ponomarev, "Caterpillar: A business process execution engine on the Ethereum blockchain," *Softw., Pract. Exper.*, vol. 49, no. 7, pp. 1162–1193, Jul. 2019.

[40] F. Corradini, A. Marcelletti, A. Morichetta, A. Polini, B. Re, and F. Tiezzi, "Engineering trustable and auditable choreography-based systems using blockchain," *ACM Trans. Manag. Inf. Syst.*, vol. 13, no. 3, pp. 1–53, Sep. 2022.

[41] M. Autili, A. Di Salle, F. Gallo, C. Pompilio, and M. Tivoli, "CHOReVOLUTION: Service choreography in practice," *Sci. Comput. Program.*, vol. 197, Oct. 2020, Art. no. 102498.

[42] F. Loukil, K. Boukadi, M. Abed, and C. Ghedira-Guegan, "Decentralized collaborative business process execution using blockchain," *World Wide Web*, vol. 24, no. 5, pp. 1645–1663, Sep. 2021.

[43] Q. Lu, A. Binh Tran, I. Weber, H. O'Connor, P. Rimba, X. Xu, M. Staples, L. Zhu, and R. Jeffery, "Integrated model-driven engineering of blockchain applications for business processes and asset management," *Softw., Pract. Exper.*, vol. 51, no. 5, pp. 1059–1079, May 2021.

[44] J. G. Represa, F. Larrinaga, P. Varga, W. Ochoa, A. Perez, D. Kozma, and J. Delsing, "Investigation of microservice-based workflow management solutions for industrial automation," *Appl. Sci.*, vol. 13, no. 3, p. 1835, Jan. 2023.

[45] M. Autili, A. Perucci, L. Leite, M. Tivoli, F. Kon, and A. Di Salle, "Highly collaborative distributed systems: Synthesis and enactment at work," *Concurrency Comput., Pract. Exper.*, vol. 33, no. 6, p. e6039, Mar. 2021.

[46] (2019). *MQTT Version 5.0*. OASIS. Accessed: Apr. 27, 2023. [Online]. Available: https://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.html

[47] E. Gümrükcü, J. R. A. Klemets, J. A. Suul, F. Ponci, and A. Monti, "Decentralized energy management concept for urban charging hubs with multiple V2G aggregators," *IEEE Trans. Transport. Electrific.*, vol. 9, no. 2, pp. 2367–2381, Jun. 2023, doi: 10.1109/TTE.2022.3208627.

[48] *National Training Standard for Information Systems Security (INFOSEC) Professionals*, Standard 4011, National Security Telecommunications and Information Systems Security Committee, 1994.

[49] P. Kannisto and D. Hästbacka, "Data autonomy in message brokers in edge and cloud for mobile machinery: Requirements and technology survey," in *Proc. IEEE 27th Int. Conf. Emerg. Technol. Factory Autom. (ETFA)*, Sep. 2022, pp. 1–4.

[50] H. Pettenpohl, M. Spiekermann, and J. R. Both, "International data spaces in a nutshell," in *Designing Data Spaces: The Ecosystem Approach to Competitive Advantage*, B. Otto, M. T. Hompel, and S. Wrobel, Eds. Cham, Switzerland: Springer, 2022, pp. 29–40.

[51] H. Tardieu, "Role of Gaia-X in the European data space ecosystem," in *Designing Data Spaces: The Ecosystem Approach to Competitive Advantage*, B. Otto, M. ten Hompel, and S. Wrobel, Eds. Cham, Switzerland: Springer, 2022, pp. 41–59.

[52] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, "The many faces of publish/subscribe," *ACM Comput. Surv.*, vol. 35, no. 2, pp. 114–131, Jun. 2003.

[53] (2008). *Advanced Message Queueing Protocol Version 0-9-1*. AMQP Working Group 0-9-1. Accessed: Apr. 28, 2023. [Online]. Available: http://www.amqp.org/specification/0-9-1/amqp-org-download

[54] *Apache Pulsar*. Accessed: Mar. 23, 2023. [Online]. Available: https://pulsar.apache.org/

[55] T. Najem, "A formal semantics for supporting the automated synthesis of choreography-based architectures," in *Proc. 13th Eur. Conf. Softw. Architecture*. New York, NY, USA: Association for Computing Machinery, Sep. 2019, pp. 51–54.

[56] *The OAuth 2.0 Authorization Framework*, Standard RFC 6749, Internet Engineering Task Force, 2012.

[57] E. Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.3*, Standard RFC 8446, Internet Engineering Task Force, 2018.

[58] (2007). *Web Services Description Language (WSDL) Version 2.0—Part 1: Core Language*. Accessed: May 11, 2023. [Online]. Available: http://www.w3.org/TR/2007/REC-wsdl20-20070626

[59] *OpenAPI Specification V3.1.0*. Accessed: May 11, 2023. [Online]. Available: https://spec.openapis.org/oas/v3.1.0

[60] (2012). *W3C XML Schema definition language (XSD) 1.1—Part 1: Structures, W3C Recommendation*. Accessed: May 11, 2023. [Online]. Available: https://www.w3.org/TR/2012/REC-xmlschema11-1-20120405/

[61] (2020). *JSON Schema Specification, Version 2020–12*. Accessed: May 11, 2023. [Online]. Available: https://json-schema.org/specification.html

[62] M. Magas and D. Kiritsis, "Industry commons: An ecosystem approach to horizontal enablers for sustainable cross-domain industrial innovation (a positioning paper)," *Int. J. Prod. Res.*, vol. 60, no. 2, pp. 479–492, Jan. 2022.

[63] P. Varga, F. Blomstedt, L. L. Ferreira, J. Eliasson, M. Johansson, J. Delsing, and I. M. de Soria, "Making system of systems interoperable—The core components of the arrowhead framework," *J. Netw. Comput. Appl.*, vol. 81, pp. 85–95, Mar. 2017.

[64] *HiveMQ*. Accessed: Apr. 27, 2023. [Online]. Available: https://www.hivemq.com/

[65] W. E. Hart, C. D. Laird, J.-P. Watson, D. L. Woodruff, G. A. Hackebeil, B. L. Nicholson, and J. D. Siirola, *PYOMO—Optimization Modeling in Python*, vol. 67, 2nd ed. Berlin, Germany: Springer, 2017.

[66] A. Makhorin. *GLPK (GNU Linear Programming Kit)*. Accessed: May 9, 2023. [Online]. Available: https://www.gnu.org/software/glpk/

[67] E. Gümrükcü, A. Ahmadifar, A. Yavuzer, F. Ponci, and A. Monti, "Datafev—A Python framework for development and testing of management algorithms for electric vehicle charging infrastructures," *Softw. Impacts*, vol. 15, Mar. 2023, Art. no. 100467.

[68] (Feb. 2016). *Interoperability Maturity Model: IMM Full—Recommendations*. European Commission. Accessed: Dec. 29, 2022. [Online]. Available: https://joinup.ec.europa.eu/collection/semantic-interoperability-community-semic/document/interoperability-maturity-model

[69] M. E. Fayad and A. Altman, "An introduction to software stability," *Commun. ACM*, vol. 44, no. 9, p. 95, 2001.

[70] M. Fayad, "Accomplishing software stability," *Commun. ACM*, vol. 45, no. 1, pp. 111–115, Jan. 2002.

[71] A. AbuHassan and M. Alshayeb, "A metrics suite for UML model stability," *Softw. Syst. Model.*, vol. 18, no. 1, pp. 557–583, Feb. 2019.

[72] M. Chen, X. Wei, J. Chen, L. Wang, and L. Zhou, "Integration and provision for city public service in smart city cloud union: Architecture and analysis," *IEEE Wireless Commun.*, vol. 27, no. 2, pp. 148–154, Apr. 2020.

PETRI KANNISTO (Member, IEEE) received the M.Sc. (Tech.) degree in automation engineering from the Tampere University of Technology, Finland, in 2011, and the D.Sc. (Tech.) degree in automation engineering from Tampere University, Finland, in 2019. He joined the Tampere University of Technology, in 2009, undertaking a position as a Postdoctoral Research Fellow, in 2019, when a merger gave birth to the new Tampere University. Since 2023, he has been a Researcher with VDEh-Betriebsforschungsinstitut (BFI), Düsseldorf, Germany, and a Visiting Researcher with Tampere University, where the present work was prepared. His research interests include service architectures, interoperability, data autonomy, and data platforms in the application areas of automation, such as green electricity, mobile machinery, and process industry.

ERDEM GÜMRÜKCÜ (Graduate Student Member, IEEE) received the master's degree in electrical power engineering from RWTH Aachen University, Aachen, Germany, in 2018. He is currently pursuing the Ph.D. degree in energy flexibility management and optimization for large-scale electric vehicle charging. In 2018, he joined the Institute of Automation Complex Power Systems, E.ON Energy Research Center, RWTH Aachen University, as a Research Associate.

FERDINANDA PONCI (Senior Member, IEEE) received the Ph.D. degree in electrical engineering from Politecnico di Milano, Milan, Italy, in 2002. In 2003, she joined the Department of Electrical Engineering, University of South Carolina, Columbia, SC, USA, as an Assistant Professor, where she became an Associate Professor, in 2008. In 2009, she joined the E.ON Research Center, Institute for Automation of Complex Power Systems, RWTH Aachen University, Aachen, Germany, where she is currently a Professor in monitoring and distributed control for power systems. Her current research interests include the automation and advanced monitoring of active distribution systems.

ANTONELLO MONTI (Senior Member, IEEE) received the M.Sc. (summa cum laude) and Ph.D. degrees in electrical engineering from Politecnico di Milano, Italy, in 1989 and 1994, respectively. He started his career in Ansaldo Industria and then moved, in 1995, to Politecnico di Milano as an Assistant Professor. In 2000, he joined the Department of Electrical Engineering, University of South Carolina, USA, as an Associate Professor, and then a Full Professor. Since 2008, he has been the Director of the E. ON Energy Research Center, Institute for Automation of Complex Power System, RWTH Aachen University. He is the author or a coauthor of more than 300 peer-reviewed articles published in international journals and in the proceedings of international conferences. He was a recipient of the 2017 IEEE Innovation in Societal Infrastructure Award. He is an Associate Editor of IEEE Systems Journal and *IEEE Electrification Magazine*, a member of the Editorial Board of *SEGAN* journal (Elsevier), and a member of the Founding Board of the *Energy Informatics* journal (Springer).

SAMI REPO received the M.Sc. (Tech.) degree in electrical engineering and the D.Sc. (Tech.) degree in electric power engineering from the Tampere University of Technology, Finland, in 1996 and 2001, respectively. He has been a Professor in electrical engineering with Tampere University, since 2011. His research interests include the various areas of green electricity and energy, including but not limited to the integration of distributed generation to electricity distribution networks, active network management, the application of AMR data in network management, the application of common information model (CIM) and IEC 61850, coordinated voltage control, and protection system management.

DAVID HÄSTBACKA (Member, IEEE) received the M.Sc. (Tech.) and D.Sc. (Tech.) degrees in automation science and engineering from the Tampere University of Technology, Tampere, Finland, in 2007 and 2013, respectively. He is currently an Associate Professor with Tampere University, Tampere. His research interests include system and software architectures and the interoperability of software systems in production and energy systems applications.