Decentralized Service Platform for Interoperable Electro-Mobility Services Throughout Europe

Nils Masuch[1(✉)], Elif Eryilmaz[1], Tobias Küster[1], Udo Pletat[2], Johannes Fähndrich[1], Thodoris Theodoropoulos[3], Mariza Koukovini[4], Natalia Selini Hadjidimitriou[5], and Nikolaos Dellas[4]

[1] DAI-Labor/Technische Universität Berlin, Ernst-Reuter-Platz 7, 10587 Berlin, Germany
{nils.masuch,elif.eryilmaz,tobias.kuester, johannes.faehndrich}@dai-labor.de
[2] IBM Deutschland Research and Development GmbH, Schönaicher Str. 220, 71032 Böblingen, Germany
pletat@de.ibm.com
[3] Institute of Communication and Computer Systems (ICCS), Iroon Politechniou 9, 15773 Zografou, Greece
thodoris.theodoropoulos@iccs.gr
[4] SingularLogic, Achaias 3 & Trizinias, 14564 Kifisia, Greece
mariza.koukovini@gmail.com, nikolaos.dellas@gmail.com
[5] ICOOR/UNIMORE, via Amendola 2, 42122 Reggio Emilia, Italy
selini@icoor.it

Abstract. While authorities are moving towards the adoption of more electro-mobility services in urban environments, their application in the real world remains still low due to a number of challenges. The integration of services from different providers using different data formats hinders interoperability; a secure and trustworthy data exchange is needed to preserve the rights of beneficiaries on the shared data. The NeMo project addresses those challenges by providing a decentralized service platform based on a distributed ledger to enable interoperable and secure data flow between service providers throughout Europe. This paper provides details about the architecture, enabling easy integration of electro-mobility services and demonstrates the approach by showing the overall flow from designing to using the electro-mobility services.

Keywords: Electro-mobility · Service development · Interoperability · Decentralized architecture · Distributed ledger

1 Introduction

The market of electric vehicles has been steadily increasing for the last years with significant growth for Battery Electric Vehicles (BEVs) in 2018 compared

B. Müller and G. Meyer (Eds.): *Towards User-Centric Transport in Europe 2*, LNMOB, pp. 184–199, 2020.
https://doi.org/10.1007/978-3-030-38028-1_13

to 2017 in Europe (cf. [3]). The resulting increased relevance of electro-mobility and the approach of digitizing products leads to a demand for custom-fit services. Especially in the context of electric-mobility a variety of stakeholder such as vehicle manufacturers, charge point operators or navigation service provider need to provide data and services in order to guarantee a smooth BEV experience for drivers and owners. However, there is not a common repository where it is possible to find them, and the provided solutions are not interoperable per se. In [16], it is pointed out that electro-mobility services are addressed by a variety of actors but their integration has not been considered, yet, which leads to a limitation in setting up innovative and interconnected services.

A typical complex electro-mobility service is to assist the electric vehicle (EV) driver in planning appropriate charging intervals at available charging stations according to its mobility needs and vehicle characteristics. In order to realize such a service, multiple information sources from potentially different stakeholders have to be combined. First, there is a need for a service that estimates the user's trip intentions based on learned user patterns for a specific time interval in the future. This service can e.g. be provided by a navigation service provider. Next, a service is necessary that computes the State-of-Charge (SoC) behaviour of the vehicle throughout the day, deduces charging needs, and provides charging strategies which would be optimal with respect to the mobility demand and vehicle characteristics. As the latter parameter might only be known by the vehicle manufacturer, this service can only be provided by this stakeholder. Finally, it is necessary to evaluate which charging stations are available and which one would be optimal according to the route. Therefore, a charge point operator service is needed that retrieves all charging stations and filters the suitable stations together with the computed charging strategy to the EV driver.

In order to realize this complex service, several technical and architectural challenges have to be overcome with respect to interoperability, security and decentralization, just to name a few. This paper presents the technical architecture of an electro-mobility marketplace that has been developed within the H2020 project *Hyper-Network for electro-Mobility* (*NeMo*)[1] and addresses the aforementioned challenges. Since platforms usually have the disadvantage that one or a few stakeholder control the system, the NeMo Hyper-Network consists of **a network of nodes** each providing the core components for developing, trading and executing IT services, which also means that there is no single point of failure. With respect to the aspect of interoperability, we decided to solve this issue by creating **a Common Information Model (CIM)** that provides a comprehensive uniform data model for representing a rich variety of concepts relevant in the electro-mobility domain. This guarantees a homogenous environment and can be a starting point for standardization efforts. Furthermore, NeMo provides **a marketplace** where partners can provide, find and integrate electro-mobility services as well as **an execution environment** for enabling a standardized interaction of the services owned by project partners.

[1] NeMo Project: https://nemo-emobility.eu.

The rest of the paper is structured as follows. Section 2 describes the architecture of the NeMo Hyper-Network. In Sects. 3, 4, 5 and 6, the details of the components presented in the architecture that have been developed for the creation, delivery and execution of complex electro-mobility services are provided. Section 7 shows, how the work presented in this paper distinguishes itself from other current approaches. Section 8 concludes the paper with final remarks.

The NeMo project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no 713794.

2 Architecture

The high-level architecture of the NeMo Hyper-Network consists of a network of equivalent nodes, which are operated by the project participants and enable them to trade and execute electro-mobility services developed with the service creation tools provided by the project. Figure 1 shows the variety of market participants in the growing space of electro-mobility, including among others drivers and owners of electric vehicles, vehicle manufacturers, navigation service providers, road operators, energy providers, and charge point operators.

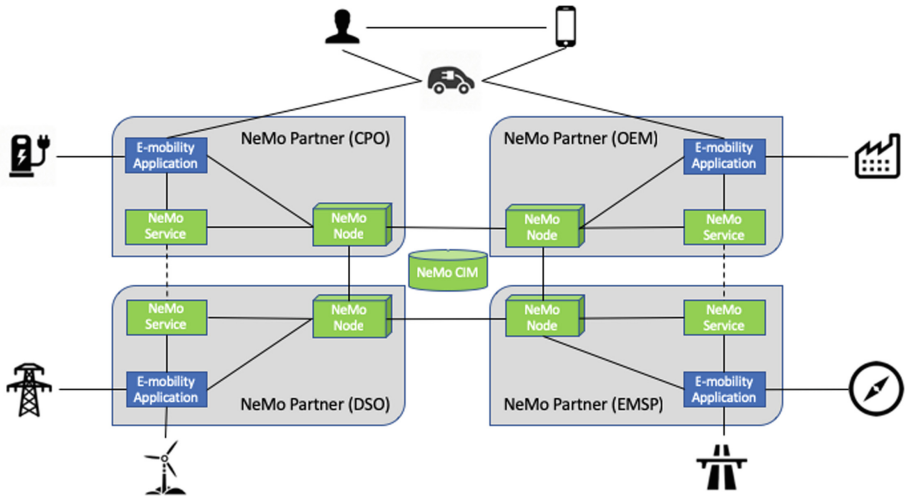

Fig. 1. A scenario with multiple organizations interacting via the NeMo Hyper-Network

The design of this network has been driven by the objective to be fast and scalable, to reduce necessary communication over the network, and to be robust w.r.t changes or problems in the network infrastructure. To achieve those objectives, the system has been designed to be decentralized, in order to prevent a single point of failure or communication bottleneck and to limit the communication between the nodes, in the extreme case allowing users to interact only

with their local nodes, while still having access to all services provided by other nodes, if needed. At the same time, this gives each participant in the network both, sovereignty over their own services that they provide to other partners, and control over the services they themselves use.

To this end, each node consists of the same components, belonging to three domains addressing the core capabilities enabled by NeMo:

- **Service Creation:** everything to create and deploy a service to the node of a partner (cf. Sect. 4),
- **Service Delivery:** everything to register and trade services on the marketplace (cf. Sect. 5),
- **Service Execution:** everything to perform service-to-service invocations within and across the nodes of the network (cf. Sect. 6).

Figure 2 illustrates the core components available on each node; they will be described in more detail in subsequent sections.

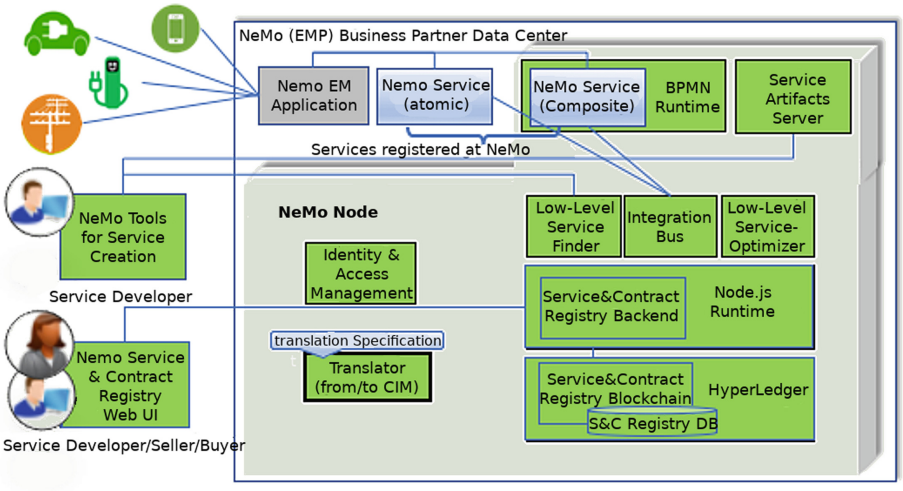

Fig. 2. The components deployed to a NeMo node

On each node, the *Service & Contract Registry* and the *Integration Bus* are the two components enabling the cross-node interactions in the NeMo Network. The Hyperledger underlying the Service & Contract Registry replicates all master data between the nodes and assures data consistency. The Integration Bus enables the service-to-service invocations across nodes.

This distributed architecture assures that the network can grow and shrink as members join or leave the consortium. New members will get their node and have full access to the marketplace as if they had been members from day one on; members leaving the consortium will loose their right to access the marketplace in

a controlled manner. In the provided architecture, each participating node can be sized according to the workload that the owner of the node causes. Additionally, in the NeMo architecture, there is no single point of failure: when a network node is not available, the remaining nodes can still operate as usual.[2]

Relating back to the use case scenario from Sect. 1, the involved services may need to be executed in different nodes as they are developed by different service providers, but at the same time need to have e.g. a uniform definition for the required parameters. The NeMo Network is implemented based on a mix of established state-of-the-art technology combined with several advanced features. The next Sections describe several innovative aspects of implementing the NeMo Hyper-Network.

3 Common Information Model

The Common Information Model (CIM) forms the base language of service creation and execution within the NeMo Network. It promotes interoperability of electro-mobility services and enables user experience by providing an interoperable format for service execution and legacy service interfacing. The electro-mobility sector can benefit from the interoperability of relevant services offered, including charging, paying, finding and reserving available EV charging stations.

Existing or new services are built using the CIM whereas data translators enable the translation of data from other proprietary formats of legacy services to the CIM. This process supports the interoperability of electro-mobility web services since it provides a common interface for web service invocation and specification. The CIM spans a wide range of electro-mobility related objects such as the electric vehicle, charging infrastructure, the EV driver, grid architecture, smart charging functionalities, and grid objects. It also supports marketplace business objects relevant for the trading of e-mobility services, like business partners, employees, services, service offerings, and contracts. The specification of CIM is publicly available online.[3]

4 Service Creation

The domain of service creation comprises of three main topics: (1) finding and reusing existing services, (2) managing services and orchestrating them to complex processes using BPMN (Business Process Model and Notation), and (3) defining translations between data formats used by individual services and CIM. Those three main topics are explained in the following.

[2] In the prototypical implementation, the Hyperledger network uses only a single instance of the so-called 'Orderer' service which is relevant for the propagation of transactions within the network and the necessary consensus-finding. However, more complex configurations of the Hyperledger infrastructure (using Kafka for the communication between the Hyperledger nodes) allows for eliminating this risk.

[3] CIM Specification: http://nemo-emobility.eu/wp-content/uploads/sites/10/2019/03/NeMo_CIM.zip.

4.1 Service Finder and Optimizer

The *Service Finder* provides two main functionalities in NeMo. First, at design-time, developers who want to reuse existing services can specify what they are looking for, and the best-fitting services will be found, sorted, and filtered to the developer's needs. Second, at run-time, one may need to find equivalent service to replace services that are no longer available or are malfunctioning, which have to take the same input and produce the same output and effects.

The Service Finder takes as input a semantic service description template in the format of OWL Web Ontology Language for Services (OWL-S)[4] instance, which besides a textual description and a name also includes Input, Output, Precondition and Effect (IOPE). Details on how the semantic service matcher works can be found in [4].

At design-time, the finder is integrated into the Semantic Service Manager (SSM) (see Sect. 4.2). Here the user can specify the template of a service that should be found and use the returned result. Specifically for NeMo, additional filter methods, e.g., for filtering by the service provider, have been implemented to ease the use of the service finder.

At run-time, the finder is called automatically from the BPMN Runtime (see Sect. 6.1) if a template instead of an actual service is provided in a step of the process. In this case there is no mechanism to handle different inputs or outputs, thus only exact matches can be allowed during run-time, excluding fuzzy semantic similarities like [6] used during design-time.

The *Service Optimizer* component has been developed for optimizing the search results retrieved from the Service Finder based on non-functional criteria. Relevant Quality of Service (QoS) parameters have been identified for electro-mobility services based on ISO/IEC 13236[5] and W3C Web Service Architecture (WSA)[6] standards. Those QoS parameters are *average availability* of a service in percentage and *average response time* of a service in milliseconds. Service providers have to agree to provide those parameters when registering their services to the NeMo Hyper-Network.

The *Service Optimizer* provides functionalities for (1) sorting the services retrieved from the Service Finder based on one QoS parameter; (2) optimizing the search results based on multiple identified QoS parameters. For the (1), the sorting depends on the type of the QoS parameter e.g. for the average availability, it sorts from high to low, for the average response time, it sorts from low to high. For the (2), Weighted Sum Model (WSM) has been implemented which is a commonly used in multi-objective optimization method [17]. By using WSM model, the following maximization function is defined to calculate the score:

[4] OWL-S: Semantic Markup for Web Services: https://www.w3.org/Submission/OWL-S/.

[5] ISO/IEC 13236 Telecommunications and information exchange between systems standard: https://www.iso.org/standard/27993.html.

[6] W3C WSA standard: https://www.w3.org/TR/ws-arch/.

$$\underset{s}{\text{maximize}} \sum_{i=1}^{n} w_i x_i(s).$$

This function assumes that there are quality characteristics x_1 to x_n which can be derived from a service s and weights for these quality characteristics w_1 to w_n. In NeMo, service providers provide the average QoS values while registering their service to the NeMo Hyper-Network. The goal is to find a service that maximizes the weighted sum of these quality characteristics and sort them based on this score (from high to low). For NeMo services, average response time which is supposed to be minimized, is combined with a negative weight, to make it a maximization goal. In NeMo, we have used two QoS parameters to calculate the score with different customizable weights.

To use different-scaled QoS parameters in the WSM, the QoS parameters need to be normalized to become in the same scale. For this purpose, different normalization techniques have been implemented, using statistical methods (e.g. *tanh*, sigmoid, z-score, logistic regression and *pnorm*). In WSM, the types of different quality parameters to maximize or to minimize are also taken into account by using positive or negative weights, respectively.

4.2 Service and Process Modelling

The Service Finder and Optimizer are integrated into a set of tools as extensions to the Eclipse IDE[7] for creating and searching services, managing available services, and orchestrating those services to more complex processes (cf. Fig. 3). Those tools are connected with each other and with other parts of the development environment and execution platform: service descriptions can be automatically derived from Java methods and vice versa, as well as deployed to and imported from a file server; semantic services and service plans can be imported into a process, the process can be deployed to the BPMN Runtime, and its execution can be monitored in the editor.

The *Semantic Service Manager (SSM)* is used to describe the services in OWL-S [13]. It provides means for creating and managing OWL ontologies and OWL-S service descriptions directly within Eclipse, and to link those to the actual implementations, e.g. deriving OWL classes from Eclipse Modelling Framework models (Ecore)[8] and OWL-S services from Java methods. It uses the Service Finder and Optimizer for finding services with the best QoS. A semantic service planner, based on a state-space planner adapted for use with OWL-S services, can be used to automatically derive service chains leading from some starting state to a goal state (all described in OWL) [5], provided that the service descriptions are sufficiently detailed, i.e. specify preconditions and effects.

Those OWL-S service descriptions have to be hosted on a web server in order to be accessible and usable by other services. For this, a special *OWL-S File Server* has been created. Besides providing an Application Programming

[7] Eclipse IDE, https://www.eclipse.org/ide/.
[8] Eclipse Modeling Framework (EMF) https://www.eclipse.org/modeling/emf/.

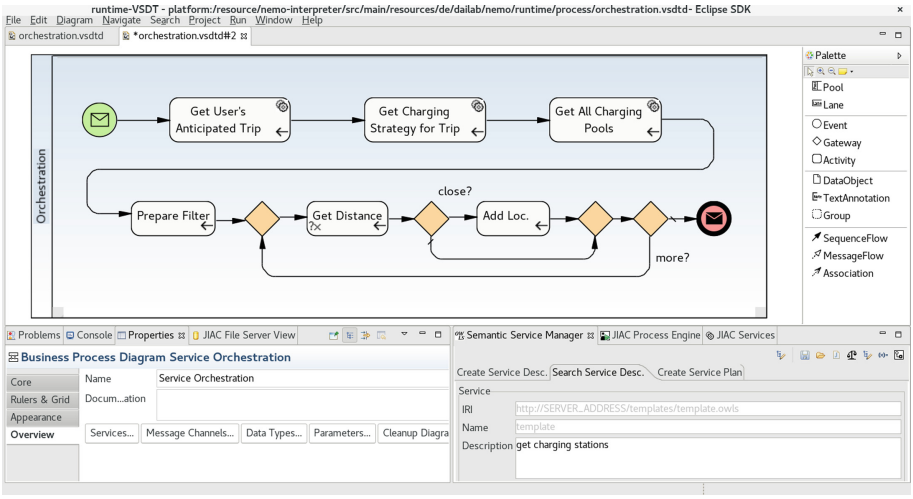

Fig. 3. VSDT (top) and SSM (bottom right), showing the example process of getting available charging stations close to the user's destination.

Interface (API) for browsing, adding or removing OWL-S description, its key feature is to update OWL-S files when they are uploaded to the file server, automatically adjusting the URLs within the OWL-S XML file to the new location on the server. The file server can be used with different back ends, either storing files directly on the computer hosting the server, or using the GitLab API[9] to store them in a GitLab repository.

Those basic and application-specific services are then orchestrated to complex processes using the BPMN editor *VSDT (Visual Service Design Tool)* [10]. While the VSDT and its underlying model are heavily based on BPMN [14], they feature a number of adaptations targeted at modeling and execution of dynamic processes, using both synchronous and asynchronous messages (service calls and multicast-messages), semantic services, and a simple expression language that can be translated to several concrete target languages. BPMN processes created with the VSDT can be translated to Business Process Execution Language (BPEL) (not used in this project), or executed by JIAC agents [12]. The VSDT also provides a simulation component, which can be used for debugging the processes prior to deployment. An extension of this component has also been used as the basis for the BPMN Runtime, which will be discussed in Sect. 6.

4.3 Data Translator

Data translators working on the basis of the CIM constitute the main interoperability enablers within NeMo, and as such have a dual role: (1) transform messages between platform specific data models used as input and output data

[9] GitLab API: https://docs.gitlab.com/ee/api/.

of registered services and the CIM, and (2) appropriately modify them, where necessary, in order to ensure security and privacy compliance. The former functionality concerns service providers that wish to seamlessly offer their services through NeMo, while the latter provides for mediating the communication of both consumers and providers with the NeMo Hyper-Network, in order to effectively protect their resources. The Data Translator life-cycle comprises two main phases as configuration and execution.

Phase I-Configuration: Before a service can be offered, the Data Translator instance deployed at the node of the Service Provider (SP) owning the service must be configured accordingly so that this service can be made seamlessly accessible according to both NeMo policies and SP data visibility preferences. This configuration will take place right before the registration of the service to the *Services and Contracts Registry*.

During this procedure, the Data Translator is provided with the *PSM-to-CIM mapping* (PSM: platform-specific model), which reflects the exact relationships between the platform-specific data model based on which the service is implemented and the CIM, in both semantic and syntactic terms. Further, this mapping includes SP-defined policies for data handling, by allowing the SP to specify visibility levels for all data involved at a fine-grained level. Based on this information, the Data Translator internally generates the logic required to transform incoming requests and outgoing responses at invocation time, expressed in the Transformation Workflow Language.

The logic generated through the above, referred to as *translation workflow*, is essentially a description of well-specified steps that will guide the translation of CIM-based request messages to PSM requests and, conversely, PSM-based responses to CIM messages. These steps are functions that are performed at run-time (i.e., during service invocation) and are derived through combining four factors: the specific transformation needs, as implied by the mapping; the security and privacy needs, established through proper queries to NeMo's policy-based framework; the SP's service policy; the available transformation functions.

Phase II-Execution: At service invocation time, the orchestration of transformation functions specified during configuration will be executed on demand, i.e., on each service request/response. In this context, the functionality of the Data Translators will essentially be grounded on the proper utilization of suitable processing components, that fall under two categories. First one is the Translation Enablers including methods for the transformation of data from the CIM to each participant's language and vice versa. Second is the Security and Privacy Enablers constituting a NeMo-wide affluent library of software tools comprising legacy cryptographic mechanisms based on the Java Cryptography Architecture (JCA),[10] as well as other custom data reduction methods tailored to the project's needs and requirements.

[10] Java Cryptography Architecture: https://docs.oracle.com/javase/8/docs/technotes/guides/security/crypto/CryptoSpec.html.

5 Service Delivery

The *Service Delivery* domain represents the distributed marketplace for e-mobility services that the NeMo Hyper-Network provides, thus following and contributing to a trend that the Blockchain approach suggests, see, e.g., also [9] where concepts of an Ethereum based distributed marketplace are described. Centralized marketplaces are usually implemented on the basis of e-commerce platforms owned and operated by a single organization leading to a 'hub-and-spoke' network as pursued in the Green eMotion[11] project whereas NeMo prefers the 'peer-to-peer' approach.

The Service Delivery domain is given by the *Service & Contract Registry* where all information about the NeMo Marketplace is maintained. The registry is therefore based on the 'marketplace business objects' subset of the CIM. This subset defines all entities to represent the organizations participating in the NeMo network, the users of the marketplace as well as the services, the offerings and the contracts signed between service providers and service consumers acting on an electro-mobility service marketplace, see also Sect. 3.

The users of the registry are developers of services registering them on the marketplace; offering managers can define the commercial terms and conditions for consumers who intend to purchase a service; finally, services can be contracted by a consumer thus acquiring the right to use a service. These Marketplace transactions are initiated on the node of a respective member of the project consortium and the results are replicated to all nodes in the network according to the underlying Hyperledger technology, see also below.

The *Service & Contract Registry* is a web application implemented using the Angular 2[12] and the Loopback[13] frameworks for the end-user functionality provided by the NeMo Marketplace. These two frameworks represent state-of-the-art technology for crafting the user and application programming interfaces of sophisticated web applications.

The core of the *Service & Contract Registry* is its Hyperledger[14] based persistence layer which assures that transactions affecting the marketplace 'world state' can be executed locally on a respective node and that all relevant marketplace data is consistently replicated among all nodes within the network.

Hyperledger implements a so-called 'permissioned Blockchain' approach where the participants of the business network are known to each other and implement agreed-upon business logic in so-called *smart contracts* [8]. All transactions initiated at one node of the network are subject to a consensus mechanism controlled by Hyperledger. This assures that only transactions to which a majority of network nodes can agree are accepted and then replicated to all nodes.

This 'permissioned Blockchain' approach and the data replication and consensus algorithms underlying Hyperledger Composer/Fabric fit very well with

[11] Green eMotion Project: http://www.greenemotion-project.eu.
[12] Angular 2: https://angular.io.
[13] Loopback: https://loopback.io.
[14] Hyperledger: https://hyperledger.github.io.

the objective to form a business consortium where each consortium member operates (or has access to) a node to participate in the marketplace and performs transactions that are accepted and agreed upon by all market participants.

6 Service Execution

The domain of service execution mainly comprises the BPMN Runtime, responsible for executing the complex processes, and the Integration Bus, which provides the unified interface for all the services in the NeMo domain.

6.1 BPMN Runtime

The processes are interpreted by the BPMN Runtime component, which is based on the process interpreter for the JIAC framework [10] with several extensions for integration into the NeMo architecture, in particular for searching for services and for invoking service via the Integration Bus (cf. Sect. 6.2). The BPMN Runtime provides support for, among others, complex conditions and looping behaviours, synchronous and asynchronous messaging, event-handling (based on incoming messages, time and duration, and errors), as well as automated semantic service planning at run-time (experimental).

The BPMN Runtime provides several web services for deploying new processes and managing existing ones. Also, each BPMN process that is deployed to the BPMN Runtime is automatically exposed as a web service and can be invoked directly or via the Integration Bus. This creates a new interpreter instance, being independent of and isolated from earlier or concurrent executions of the same or other processes. The process is executed, one task at a time, by the agent's execution cycle. This way, even with multiple processes running in parallel, speeds up to 100 cycles per second (i.e. an execution interval of 10 ms) are no problem.

While BPMN provides many task types, in the context of NeMo the most important type is the Service task, which triggers the invocation of another service within the process. Depending on the type of service being called – process, basic service, or service template – the service will either be called directly, or via the Integration Bus. In the case of service templates, at first a service matching the template is searched by calling the Service Finder and Optimizer via the Integration Bus, before invoking the actual service (if any was found).

6.2 Integration Bus

The Integration Bus is the central component for executing intra-node and cross-node service-to-service calls. It exposes a public REST API to client applications or services to initiate service calls. Such service calls initiated at a source node can (1) request an atomic service on local or remote target node, (2) request a composite service on local or remote target node, or (3) search for services.

The Integration Bus uses an internal REST API to propagate calls from its *Integration Bus-Source* component to its *Integration Bus-Target* component.

The location of the target service to be invoked determines whether the call is sent to a remote node or whether it remains on the local node. Figure 4 illustrates the invocation paths enabled by the Integration Bus.

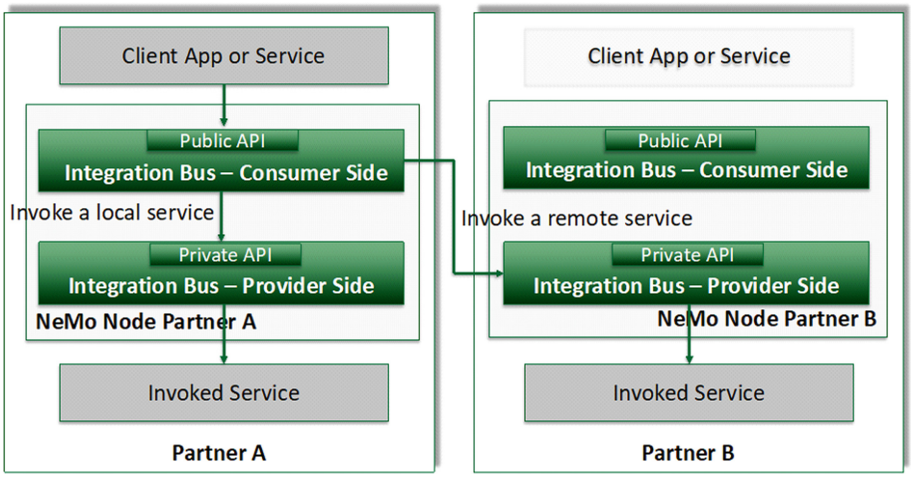

Fig. 4. NeMo Integration Bus service invocation paths.

During an integration bus transaction the consumer side performs a lookup of the service within the service registry. Once the service details are loaded from the service registry, the transaction is routed towards the consumer side. Within the consumer side of the integration bus the following operations take place: Initially the transaction is translated from the Common Information Model format to a proprietary format by the data translator. Subsequently the transaction is forwarded to the Service Invoker which invokes the required service. This Invoker represents the last mile on the target node where a service is being invoked; it is available for calling services exposed via a REST or SOAP API. Finally the resulting data object of the web service is translated back to the CIM and it is returned to the client side of the integration bus and the client app.

6.3 Workflow

The general workflow will be explained by coming back to the example from the introduction. The example process can be seen in Fig. 3, whereas Fig. 5 shows a sequence diagram of the general workflow.

First, a user would use the search capabilities of the Integration Bus to look for a service or process matching their use case. Let's assume that the user has found the example process. They would then invoke that process via the Integration Bus, which will lookup the grounding of that service (since the process is a service, too) and invoke the corresponding REST service provided by the

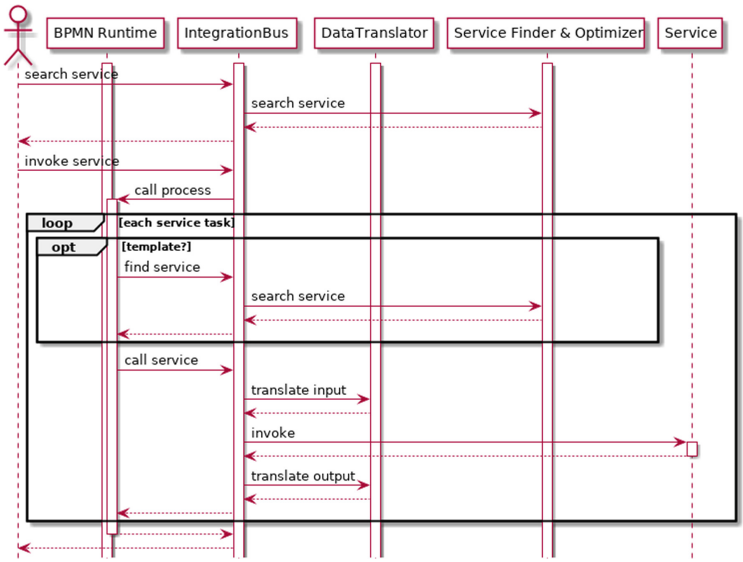

Fig. 5. Sequence diagram of the workflow, slightly simplified.

BPMN Runtime with the parameters provided by the user, which will then be creating a new interpreter instance. The interpreter steps through the BPMN process and soon encounters the first service, for getting the user's planned trip. Now, the BPMN Runtime will invoke the Integration Bus itself, which again looks up the grounding of that service and invokes the basic service. This will trigger the Data Translator, translating from the CIM model to the platform-specific model of that service, and back after the service returned a result. If the service description in the Runtime is a template, it will first query a matching service from the Integration Bus. The result is passed back to the BPMN Runtime (here, no translation is necessary, as the process works directly on the CIM). The interpreter continues with the next two services, which are invoked in the same manner. Finally, it executes the remaining process in the lower half of the BPMN pool, working on the data retrieved from the former three service invocations and aggregating the result, which is a list of charging stations close to the user's destination. Finally, the result is returned to the Integration Bus and thus to the user.

7 State of the Art

Several electro-mobility services are already available in the market. As mentioned in the introduction, a common repository for their discovery is missing and at the same time the interoperability issue remains. This is emphasized in [16], where it is stated that the integration of different eleoctro-mobility services has not been considered comprehensively so far. For this reason, the

authors provide a framework to identify the characteristics of e-mobility services and conclude that, besides the categorization of the services, it is necessary to describe the relationships between the different market players. The interoperability between electro-mobility services has been considered by [2] who present the results of the Internet of Energy (IoE) for Electric Mobility project, which consisted of the development of hardware, software, and middleware solutions to create an inter-operable communication infrastructure between stakeholders of the smart grid. Information and Communications Technology (ICT) platforms for overcoming interoperability issues have been already proposed in the past, but, to the authors' best knowledge, there is no real working example. The Green eMotion project,[15] for instance, proposed an ICT Reference Architecture and Business-to-Business (B2B) marketplace based on which the NeMo project has been developed. In a sense, NeMo addresses the trend towards more decentralized and 'democratic' marketplaces where all participants operate their share of the marketplace (enabled through the Blockchain approach used by NeMo). Moreover, NeMo combines service creation, marketing and execution in a single platform, whereas earlier approaches as mentioned above tend to handle these aspects separately.

The European Green Cars Initiative-Public Private Partnership was instead focused on novel architectures, testing concepts, and standardization needs for EVs. Another European project, MOBINET,[16] studied the technical and organizational foundations of an open multi-vendor platform for European mobility services.

The interoperability among cross-border charging infrastructure was demonstrated in the context of the CROME project as described in [7], which has developed a marketplace for the exchange of data and for roaming between different ICT systems based on the Bosch brokering platform. Similarly to NeMo Hyper-Network, the CROME platform allows the partners to run their systems autonomously but its functionality is limited to the charging infrastructure, while NeMo allows to create different types of electro-mobility services and provides the possibility to create new services on top of existing ones. A comparison of mobility services marketplaces is presented by [15]. A few platforms provide e-mobility services but none of them implement inter-connectivity because no protocol provides the possibility to access the internal data that are important for the customer and for the roaming. The Chargepoint[17] and Open Charge Map[18] projects, for instance, provide information on the charge points but they are not able to exchange data between each other. In NeMo Hyper-Network, instead, data are owned by the provider but can be used to create new services.

In [11] the interoperability of electro-mobility services is tackled by proposing an ICT architecture with standardized communication protocols for charge point

[15] Green eMotion: Development and demonstration of a unique and user-friendly framework for green electro-mobility in Europe. FP7-TRANSPORT.
[16] Europe-Wide Platform for Cooperative Mobility Services. FP7-ICT.
[17] Chargepoint: https://www.chargepoint.com.
[18] Open Charge Map: https://openchargemap.org.

management. In this regard, NeMo considers the Open Charge Point Protocol (OCPP) as the only standard broadly rolled out for the coming years [1].

8 Conclusion

This paper described the results of the NeMo project, providing an architecture for implementing, describing, finding, orchestrating, and deploying electromobility services in a decentralized, heterogeneous, vendor- and platform-independent way. When the project started, it aimed to contribute a number of advances to the state-of-the-art, which are still valid now, as it comes to an end:

- The Common Information Model is a basis for interoperability between e-mobility services as it standardizes the data exchanged between services.
- The marketplace allows for trading services addressing the needs of e-vehicle charging, trip planning, vehicle state analysis, etc. There is no conceptual limitation on the functionality of services being traded.
- Numerous innovative e-mobility services have been implemented and are available as starting points for growing this marketplace.
- Finding services on the marketplace is based on semantic search capabilities, which go beyond typical 'syntactic only' searching.
- NeMo offers a decentralized service trading and service execution platform available to all project participants.

To enable the above points, the parts of the architecture have been implemented by different partners in the project consortium, integrated, and evaluated using a number of e-mobility services and service orchestrations, showing promising results. A large-scale roll-out of the NeMo Hyper-Network is in progress, and shall be further pursued by the NeMo Association after the end of the project. The goal of this second phase will be to integrate a critical mass of stakeholders and thus increase the number of relevant services provided by the NeMo Marketplace. At the same time, it is important that the elaborated business concepts for the NeMo Hyper-Network are well incorporated into the technical approach in order to provide a market-ready solution. Given that, new and innovative solutions will be provided to the end user and will hopefully make the application of electric mobility in daily life even more attractive.

References

1. Albertus, S., Castagnie, A.: NeMo D1.2 report on latest technological and market developments in electromobility. Technical report, Renault (2017)
2. Bedogni, L., Bononi, L., Di Felice, M., D'Elia, A., Mock, R., Montori, F., Morandi, F., Roffia, L., Rondelli, S., Cinotti, T.S., Vergari, F.: An interoperable architecture for mobile smart services over the internet of energy. In: 2013 IEEE 14th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM), pp. 1–6 (2013)

3. Bratzel, S.: E-mobility 2019: an international comparison of important automotive markets. Consolidated sales trends for full-year 2018 and forecast for 2019. Technical report, Center of Automotive Management, January 2019
4. Fähndrich, J., Masuch, N., Borchert, L., Albayrak, S.: Learning mechanisms on OWL-S service descriptions for automated action selection. In: IoA17 at 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS), pp. 56–73 (2017)
5. Fähndrich, J., Masuch, N., Yildirim, H., Albayrak, S.: Towards automated service matchmaking and planning for multi-agent systems with OWL-S – approach and challenges. In: Service-Oriented Computing – ICSOC 2013 Workshops. LNCS, vol. 8377, pp. 240–247. Springer (2014)
6. Fähndrich, J., Weber, S., Ahrndt, S.: Design and use of a semantic similarity measure for interoperability among agents. In: Proceedings of 14th German Conference on Multiagent System Technologies (MATES 2016), pp. 41–57. Springer (2016)
7. Gagnol, P., Jochem, P., Pierre, M., Fichtner, W.: CROME: the French and German field demonstration of the interoperable mobility with EVs. World Electr. Veh. J. **6**(4), 1094–1101 (2013)
8. Gaur, N., Desrosiers, L., Novotny, P., Ramakrishna, V., O'Dowd, A., Baset, S.: Hands-On Blockchain with Hyperledger: Building Decentralized Applications with Hyperledger Fabric and Composer. Packt Publishing Ltd., Birmingham (2018)
9. Kabi, O.R., Franqueira, V.N.L.: Blockchain-based distributed marketplace. In: Business Information Systems Workshops, pp. 197–210. Springer (2019)
10. Küster, T., Heßler, A., Albayrak, S.: Process-oriented modelling, creation, and interpretation of multi-agent systems. Int. J. Agent-Oriented Softw. Eng. **5**(2/3), 108–133 (2016)
11. Lewandowski, C., Böcker, S., Wietfeld, C.: An ICT solution for integration of electric vehicles in grid balancing services. In: 2013 International Conference on Connected Vehicles and Expo (ICCVE), pp. 195–200 (2013)
12. Lützenberger, M., Konnerth, T., Küster, T.: Programming of multiagent applications with JIAC. In: Industrial Agents – Emerging Applications of Software Agents in Industry, chap. 21, pp. 381–400. Morgan Kaufmann, Boston (2015)
13. Masuch, N., Kuster, C., Albayrak, S.: Semantic service manager – enabling semantic web technologies in multi-agent systems. In: Proceedings of Joint Workshops on Semantic Web and Big Data Technologies, INFORMATIK 2014. LNI, GI, Stuttgart, Germany, vol. 232 (2014)
14. OMG: Business process model and notation (BPMN) version 2.0.2. Specification formal/2013-12-09, Object Management Group, December 2013
15. Strasser, M., Weiner, N., Albayrak, S.: The potential of interconnected service marketplaces for future mobility. Comput. Electr. Eng. **45**, 169–181 (2015)
16. Stryja, C., Fromm, H., Ried, S., Jochem, P., Fichtner, W.: On the necessity and nature of e-mobility services - towards a service description framework. In: Nóvoa, H., Drăgoicea, M. (eds.) Exploring Services Science, pp. 109–122. Springer, Cham (2015)
17. Triantaphyllou, E.: Multi-criteria Decision Making Methods: A Comparative Study. Applied Optimization. Springer, Boston (2013)