IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 14, 2018, accepted January 4, 2019, date of publication January 17, 2019, date of current version February 8, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2893340

Internet of Smart City Objects: A Distributed Framework for Service Discovery and Composition

FIKRET SIVRIKAYA[1], NIZAR BEN-SASSI[2], XUAN-THUY DANG[2], ORHAN CAN GÖRÜR[2], AND CHRISTIAN KUSTER[1]

[1]GT -ARC gemeinnützige GmbH, 10587 Berlin, Germany
[2]DAI-Labor, Technische Universität Berlin, 10587 Berlin, Germany

Corresponding author: Fikret Sivrikaya (fikret.sivrikaya@gt-arc.com)

This work was supported in part by the German Federal Ministry of Education and Research (BMBF) under Grant 16KIS0580.

ABSTRACT Smart cities generally aim at efficiently organizing and managing city resources through a digital layer on top of the legacy infrastructure. As the digitalization trend goes on with an increasing pace and with the involvement of a diverse set of actors, proper management of this digital layer as well as the services deployed over it becomes ever more crucial. This paper presents our methodology of transforming the complex smart city concept and its digital layer into a structured model for creating a dynamic and adaptive service ecosystem in the digital cities of the future. An overview of the smart city concept and the three-tier architecture that emerges through the digitalization of cities is presented first, together with the main challenges in attaining coherent smart city service environments that can avoid fragmentation, ensure scalability, and allow reuse. The three major enablers that are identified in this direction are 1) a semantic functional description of city objects, representing physical devices or abstract services; 2) a distributed service directory that embodies available city services for service lookup and discovery; and 3) planning tools for selecting and chaining basic services to compose new complex services. For each of these, a high-level overview of the available tools and results from the research literature are provided as well as the relevant standards. This overview is complemented with our own approach and design choices in project ISCO (Internet of smart city objects). Through the implementation of two distinct use cases, this paper illustrates how ISCO components can jointly enrich service creation and consumption of various stakeholders in smart cities.

INDEX TERMS Smart cities, smart city objects, service discovery, service composition, semantic service description, adaptive planning.

I. INTRODUCTION

Smart cities emerge as a new dimension of urbanization by bridging the city infrastructure with digital services and information. The smart city concept appeals to a diverse set of actors, making digital solutions increasingly more ubiquitous in transportation, energy, healthcare, education and many other aspects of our lives. While the rapid advancements in Information and Communication Technologies (ICT) help drive this digitalization trend forward, they also create new challenges due to the increasing complexity and dynamics in the smart city service ecosystem. Service providers who want to establish cross-domain services still face the challenge of interconnecting fast-growing and diverse set of technologies. With the lack of unified data and service integration platforms, the transformation to digital cities often results in applications representing isolated service and data silos. This prevents the realization of smart city services defined to have the characteristics of being data-centric, multidisciplinary, self-adaptive, customizable to user needs and user experience [1].

To address this problem, we present our methodology of transforming the complex smart city concept and its digital layer into a common structured model for creating a dynamic, adaptive and unified service ecosystem. This is achieved by encapsulating both ICT-capable physical devices and virtual online services into a common model, referred to

as *Smart City Objects (SCO)*. A smart city object can be as simple as a temperature sensor providing data to the cloud or as complex as a trip assistance service that stems from a well-crafted composition of many other SCOs from transportation and energy domains.

The aim of this article is to highlight the major contributions of our project *ISCO (Intelligent Framework for Service Discovery and Composition*, a.k.a. *Internet of Smart City Objects*), which aims at the design, creation and holistic interconnection of smart city objects from different domains through its open and extensible platform and its supporting tools. With this objective, we move away from fragmented Internet of Things (IoT) solutions and isolated data silos in cities towards a more integrated, scalable and harmonized smart city ecosystem.

ISCO framework incorporates three major components to realize this objective. The first one consists of a city object API and supporting tools for the enriched semantic description of city objects with their functional and qualitative features. The second component is a distributed service directory that embodies available city services for service lookup and discovery, at the same time ensuring its scalability through an information-centric networking (ICN) overlay. Finally, the third component provides planning tools for selecting and chaining basic services to compose new complex services.

In the remainder of the article, we review the smart city challenges, solution approaches, and suitable tools as well as an in-depth overview of our proposed contributions in the ISCO Project. The article is organized as follows. We first provide an overview of the relevant background and related work in smart city solutions in general (Section II). We then present the ISCO approach to smart city service integration (Section III) and the overall ISCO architecture. Afterwards, the technical core of the ISCO platform is explained in detail, including a review of the technical background in relation to each presented ISCO component. We start with a unified semantic model for SCOs based on semantic web technologies (Section IV). This is followed by the scalable distributed architecture for SCO discovery (Section V). Finally, we present our two-level planning approach for orchestrating through both generic and application-specific service ontologies that also incorporate quality of service (QoS) attributes (Section VI).

II. BACKGROUND AND RELATED WORK

The smart city concept has been attracting strong attention of the research community from a large variety of research fields, particularly in the computer science and information systems disciplines [2], [3]. Although there are various definitions and numerous interpretations of the smart city concept, a common 3-layer architecture can usually be applied to the general concept, as illustrated in Fig. 1. At the lowest layer, the urban infrastructure refers to the existing physical objects in cities that could either be conventional elements like bus stops and traffic lights or potential future objects such as drones and automated ground vehicles. At the highest layer

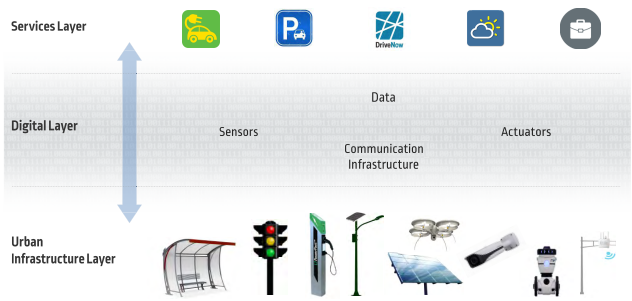

FIGURE 1. 3-tier smart city architecture, with the "digital layer" acting as a bridge between the urban infrastructure and smart city services.

of the 3-tier architecture are the smart city services, which could range, e.g., from electric vehicle charging and parking services to tourism applications and weather info services. Most crucial for a smart city is the "digital layer", which essentially interconnects and provides access to all objects in the city. This may include the digitization of various conventional elements in the city, usually by equipping them with various types of sensors, and the scalable data communication and analytics approaches for the vast amount of city data.

Advancements of future data networks and big data analytics are important enablers for the emerging IoT approaches in smart cities. Chen *et al.* [4] integrate a narrow-band communication module over an IoT framework, which enables long-distance data transmission and thus allows data sensing, computation and communication with wider coverage in urban environments. Chen and Leung [5] propose cognition-based communications, where the idea is to incorporate AI-based computing solutions to the communication layer of IoT frameworks. They offer an application in the healthcare domain through resource optimization for transmitting a user's critic health data in a city. Toward enabling the data analytics in smart cities, a big data system is proposed for improved urban healthcare through mobile crowdsourcing and IoT sensing, along with users' body signals [6]. Besides the enabling technologies, a smart city framework is crucial to incorporate existing and emerging technologies, which results in a sustainable system to address the social challenges of smart cities.

Given the vast breadth and depth of the smart city scope, we do not intend to provide a comprehensive background on specific technological enablers here, but set the stage for the remainder of the paper, which focuses on a holistic framework for smart cities. After a brief overview of the standardization efforts in the direction of smart cities, we list and compare some of the related projects in the remainder of this section. We refer the readers to the surveys on the topic, e.g., [2], [7], [8], for a more detailed coverage of the socio-technical systems of smart cities.

A. SMART CITY STANDARDS

Interoperability is one of the key ingredients of the smart city concept, as in many other complex domains with a large

and diverse set of stakeholders. International standardization plays a crucial role in this context, in order to ensure city-scale interoperability among services and devices on the one hand and transferability and reusability of solutions among different cities on the other. There is a large body of standardization groups and activities on various aspects of smart cities, the major of which are depicted in Fig. 2 in a structured form by the British Standards Institution (BSI) [9]. The standards at the top layer refer to those that guide city authorities in adopting an overall smart city strategy, including its monitoring and evaluation. The middle layer includes standardization activities that aim at the effective procurement and management of smart city projects. The bottom layer is the most relevant to the scope of this article, focusing on the standardization of technical specifications for the actual implementation of harmonious smart city solutions.

The International Organization for Standardization (ISO) is one of the most prominent actors that cover all facets of smart city standardization, as clearly observed from the figure. The CEN-CENELEC-ETSI Smart and Sustainable Cities and Communities Coordination Group (SSCC-CG) brings together various European standardization bodies and acts as a coordinating body for the European standardization activities related to smart cities. ITU Focus Group on Smart Sustainable Cities actually concluded its work in 2015, producing a number of technical specifications and reports, including the definition of a *Smart Sustainable City* [10] and the key performance indicators related to the use of ICT in smart sustainable cities [11]. A similar standardization work by ISO/IEC that is currently under development (*CD 30146 "Information technology – Smart city ICT indicators"*) is aiming to define a comprehensive set of indicators that cities can use to assess their progress in using ICT. Another standard by ISO/IEC (*Smart City Concept Model* [12]) is focused primarily on data interoperability, providing a high-level framework for describing concepts and relationships by aligning the ontologies from different sectors.

In addition to the mentioned standardization bodies, there are many other initiatives and organizations that promote the development of smart city standards. For example, the IEEE Smart Cities Initiative[1] aims to act as a neutral broker for supporting smart city technology development and standardization, although it does not currently produce any standards of its own. Furthermore some standards bodies are focusing on specific 'vertical' domains within the smart city scope, such as intelligent transport systems, smart homes, smart buildings, and eHealth.

Standardization efforts for smart cities have still a long way to go at all three layers presented in Fig. 2. Many smart city projects and initiatives around the world, some of which were presented in the previous section, are also trying to partially contribute to this endeavor. In particular, Project ESPRESSO[2] (systEmic Standardisation apPRoach to Empower Smart citieS and cOmmunities), co-funded by the Horizon 2020 Framework Programme of the European Union, has specifically focused on the identification and analysis of the gaps and overlaps across standards developed by the various standardization organizations for smart cities.

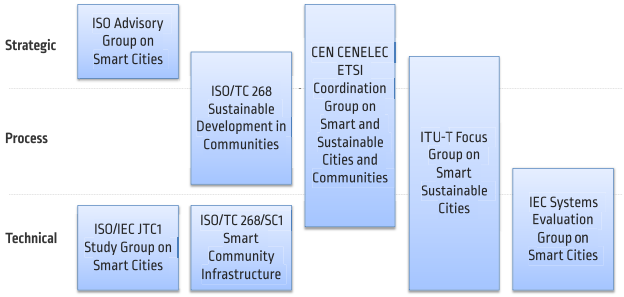

FIGURE 2. An overview of the international standardization groups working on different aspects of smart city standards.

While we do not directly target a specific standard in our smart city project ISCO, presented in the next section, we aim at providing the community with an open framework of tools and technologies that collectively address all of the required aspects of achieving vendor-agnostic compatible smart city deployments.

B. SMART CITY PROJECTS

As early as in 2011, the number of interconnected IoT objects already exceeded the world population [7]. To make use of this connectivity in bearing a positive role on the quality of the citizen's life, a growing number of smart city initiatives is spawned around the world in recent years. A report published by the European Parliament [13] in 2014 explores smart city initiatives in terms of their impact on the objectives defined in the European Growth Strategy 2020, covering more than 50 smart city projects that were conducted in 37 cities.

Among many research initiatives that deal with smart cities, the Sociotal[3] project devises a platform with an improved privacy and security aspects to encourage people in sharing their data. COSMOS[4] offers an IoT middleware focusing on the collection, labeling, processing and distribution of massive data. The project deals with the problem of generating, managing and sharing the knowledge acquired from raw sensory data. Similarly, GAMBAS[5] framework targets the data acquisition and storage but with a distributed processing of the data to understand the user behaviors and generate personalized responses fast and secure.

The diversity of the existing IoT platforms has shifted the focus of more recent projects to provide interoperability between such platforms and make them applicable in a broader sense. Some of the proposed solutions to facilitate cross-IoT-platform operations include: a modular gateway for

[1] https://smartcities.ieee.org/
[2] http://www.espresso-project.eu/
[3] http://sociotal.eu/
[4] http://iot-cosmos.eu/
[5] http://www.gambas-ict.eu/

integrating and interconnecting new devices (AGILE),[6] a virtual network layer for standard-agnostic secure communication of IoT devices from different platforms (VICINITY),[7] an interoperable IoT framework to allow the design and development of new IoT devices or services on the existing IoT ecosystems (INTER-IoT),[8] and standardized Open APIs and service creation tools to ease the introduction of new IoT solutions and to enable interoperability of them with the existing ones (bIoTope).[9] Similar to these cross-platform solutions, FIESTA-IoT,[10] which inspires our approach, provides a meta-testbed IoT infrastructure as a service to adapt and federate existing IoT platforms and testbeds. For this purpose the project offers common standards and an API to adapt the various IoT data to a common ontology.

While these projects, among many others, provide valuable contributions towards addressing the challenges of smart city IoT frameworks, their partial results still represent partial and fragmented solutions that can not be easily brought together in an efficient and scalable manner. Table 1 provides a comparative summary of the reviewed smart city projects in terms of their solution components within the aforementioned "digital layer". The middleware architecture and components for each project, i.e., modeling, orchestration, and discovery, are categorized based on a taxonomy by [14]. The respective features of our ISCO project are provided at the end of the table, contrasting its design choices and contributions with other projects.

III. ISCO APPROACH AND ARCHITECTURE OVERVIEW

ISCO Project's approach towards addressing the need for more integrated smart city solutions is to design and develop open platforms and tools for interconnecting heterogeneous entities in a city through what we call *smart city objects*. In this section, we present the overall architecture of the ISCO smart city platform and its support for smart city service workflows among different city stakeholders.

A. ISCO ARCHITECTURE COMPONENTS

The ISCO platform is the realization of the digital layer in the aforementioned 3-tier smart city architecture. As an enabler for the envisioned dynamic composition of current IoT solutions and future holistic approaches, the platform is designed to cope with both technological and socioeconomic challenges of future smart cities. In contrast to specific IoT domains, where technical solutions have central roles, solutions for cross-domain problems must also take into account the interactions and cooperation of different stakeholders who have great impact on the resulting smart city platform, i.e., city authorities, service providers, developers or technology providers, and citizens. The ISCO platform components are therefore designed to enable such

[6]http://agile-iot.eu/
[7]http://vicinity2020.eu/
[8]http://www.inter-iot-project.eu/
[9]http://www.biotope-project.eu/
[10]http://fiesta-iot.eu/

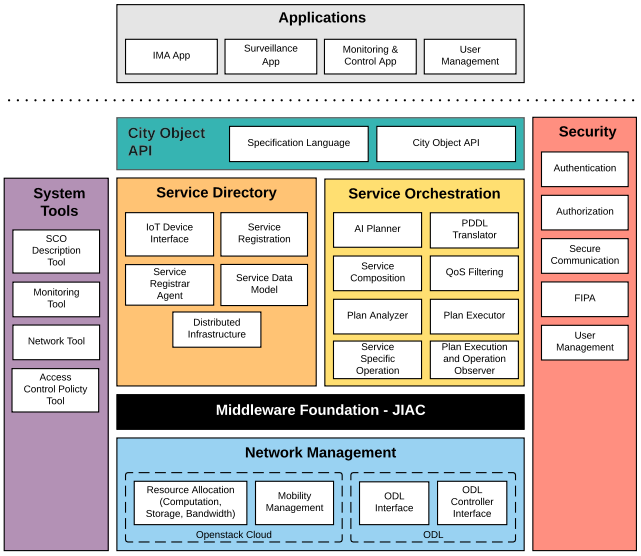

FIGURE 3. Components of the overall ISCO architecture.

multi-stakeholder interactions beside the common technological goals of scalable, dynamic and heterogeneous service platforms.

Fig. 3 shows the ISCO platform components. The focus of this article is around the components of *City Object API*, *Service Directory* and *Service Orchestration*, which are detailed later in their own sections. In the remainder of this section, we briefly describe the other components to complete the big picture of the project.

1) MIDDLEWARE FOUNDATION

At the core of the ISCO platform is a multi-agent-based middleware component. Multi-agent systems provide the desired advantages of service oriented architectures (SOA) for large-scale service orchestration. Based on semantic service descriptions, rational agents with planning capability can autonomously select own actions or cooperate with other agents to achieve their given goals. However, most of the popular agent frameworks only provide limited support for semantic web services. The ISCO middleware is realized using the JIAC agent framework [15] and extensive use of a self-developed semantic description framework for smart city services. The resulting support for semantic SCOs of the ISCO platform enables autonomous agents to represent cross-domain SCOs and realize their interactions.

2) COMMUNICATION NETWORK MANAGEMENT

The ongoing adoption of IoT deployments with billions of devices and services has a great impact on the design of the next generation mobile networks of 5G and beyond. Current IoT integration approaches that rely on mobile broadband networks, centralization of sensor data, and utilization by a single provider largely result in inefficient use of network resources. Local wireless mesh networks (WMN) could offload IoT traffic from mobile network, providing decentralized direct point-to-point (e.g, device-to-device,

TABLE 1. Comparative summary of selected smart city and IoT projects, and the respective features of the ISCO framework.

Project	Architecture	Object Modeling	Composition & Orchestration	Service Discovery
BIG-IOT	Relies on a common Restful API to connect applications, IoT platforms and the triple store MarketPlace (considered as the nucleus of its ecosystem).	Provides a semantic model that is used for the description of underlying data models: the offerings (services/ingredient), the queries (offering requests) and the recipes (service compositions).	Service compositions are semantically defined using 'Recipes'. BIG-IOT provides a user interface for the Recipe Creation and Instantiation.	The Consumer Lib allows applications or services to authenticate on the marketplace, to retrieve available offerings based on semantic queries from the triple store, and to subscribe to offerings of interest and get updates in real-time.
bIoTope	The bIoTope architecture is aligned with the IoT Architectural Reference Model (ARM), as defined in the IoT-A project.	The data/service descriptions rely on the Open Data Format(O-DF). A knowledge extraction framework is used to convert the O-DF / Open Messaging Interface (O-MI) data to RDF data. This latter is then combined with linked open datasets and queried using SPARQL.	Provides a NodeRED based user interface that allows the user to design service composition workflows. The users are also able to personalize the dashboards by aggregating information from different IoT devices in the bIoTope ecosystem	The data source discovery is carried out by the O-MI Node. The Web Service enables a bidirectional communication with the discovery and publication node using the protocols HTTP or HTTPS.
COSMOS	The COSMOS platform is based on the IoT-A reference architecture with an event based middleware.	The models are based on the reference model of IoT-A. Virtual Entities represent IoT devices and resources. The IoT services are the abstraction of these entities and resources.	Uses predefined service composition based on templated workflows. A case-based reasoning planner compares service requests with available templates by reasoning upon the provided semantic description. Planner implements MAPE-K loops.	A discovery service is provided for Virtual Entities Resolution, which is constructed around a semantic store. The Jena store is used for the semantic description of Virtual Entities and its query engine ARQ is used for the registration and lookup processes.
FIESTA-IoT	Follows a meta-cloud approach based on the IoT-A reference architecture.	Adopts the Reference Model as defined in the ARM including a Domain Model (DM), describing the different entities and their relations, an Information Model (IM), describing the structure of Virtual Entities as a representation of physical entities in the cyberspace, and a Functional Model, describing the service oriented approach.	The IoT service composer allows users to combine different kinds of sensors to generate added-value IoT services. The IoT service composition engine is responsible for executing Composite IoT Services.	For service discovery, a request is sent to the Resource Broker, which forwards the request to the Resource Manager (RM). RM is the only point of entry to Semantic Service and Resource Descriptions (SSRD) repository, which is a database that stores IoT service and resource descriptions as RDF triples.
GAMBAS	GAMBAS is a database oriented middleware. It can be viewed as a virtual IoT database system in which each component stores, processes and disseminates data.	The service models define the operator, provider and consumer roles, and the device-centric models include embedded computer systems, constrained computer systems, traditional computer systems and backend computer systems.	Data oriented interaction is facilitated by a distributed data acquisition framework, a semantic data store, and a query processing framework.	The Data Discovery Registry enables the discovery of data sources available on the network. A semantic description is used to register new data sources to the data storage. The Registry uses this semantic description to manage the incoming requests.
INTER-IoT	Multi-agent middleware with a layered approach to interconnect other IoT platforms.	Things are represented by smart objects, which are abstracted as cooperating agents characterized by their set of tasks. These tasks are event-driven, while a state-based approach is applied to model the agent's behavior and goals.	Different management components coordinate the interactions between the layers of IoT systems, e.g., task management, device management, communication management, knowledge management.	The platform is built on JADE framework, which provides an Agent Management System, a Service Manager and a Directory Facilitator. The Directory Facilitator is a centralized registry of agents and the associated smart object descriptors.
SocIoTal	Follows IoT-A reference model and builds around a Context Manager component, serving as a central database.	Users and software agents are modeled as Virtual Entities (VE) based on OMA NGSI Context model. Devices are modeled as context enablers.	The Context Manager acts as a central service broker in a publish/subscribe architecture.	The context manager serves as a centralized resource directory for all context information related to Virtual Entities and associated enablers.
VICINITY	Uses a cloud-based service oriented middleware with a central management platform.	IoT objects are modeled based on W3C Web of Things (WoT) specification. The Thing Description (TD) describes physical or abstract IoT objects, and the Things Ecosystem Description (TED) describes the integrated IoT infrastructure. Semantic representations provide the fundament of interoperability.	Central platform provides operation management, access control, and configuration management for connected things. Dynamic service composition and orchestration are not specified.	Builds on the central repository of semantic Thing Descriptions. Service registration and discovery is based on SPARQL, which is wrapped by a gateway API. Discovery uses a P2P network overlay formed by distributed gateways and central cloud components.
ISCO	ISCO is a multi-agent service-oriented middleware, tailored for distributed smart city deployments. Services and devices are represented by agents with communication capabilities.	Semantic descriptions of IoT services and devices (objects) are based on the ISCO smart city ontology, which builds on IoT-Lite and IoT-O with focus on orchestration. It covers both functional and qualitative aspects of available smart city objects.	The ISCO middleware provides an automated QoS-aware service composition engine. An extended version of the MAPE-K loop is adopted to enhance the system with adaptation capabilities.	Content-centric and distributed service directory infrastructure uses semantically annotated attributes for service lookup. Semantic object descriptions are cached among geographically distributed repositories. Efficient cache management and routing enables scalable and low-latency service discovery.

device-to-service) communication on the network edge. However, the integration and management of WMNs with heterogeneous wireless technologies, which may be provided by established and emerging network operators, pose a great challenge for the smart city platform. The ISCO platform enables composition of the network as a service with vertical city services. To realize such a flexible city data infrastructure, we employ the emerging network virtualization technologies, e.g., software defined network (SDN), network function virtualization (NFV), and intelligent approaches for autonomous network management. Abstractions of network resources, policies, and control mechanisms are incorporated in the ISCO data model, which allow application-aware orchestration and provisioning of network services.

3) SECURITY

Security and privacy have been important issues for distributed IoT solutions. This is reflected in standardized guidelines for securing gateways, communication links and systems in different IoT domains. While various security aspects, e.g., integrity, trust, authorization, can provide a secure basis for data exchange, the envisioned dynamic cross-domain interactions in smart cities require an adaptive decision making and enforcement of security policies based on relevant context information. The ISCO security component realizes such a distributed policy management framework, which infers situational context from services, sensor data, stakeholders, and their relationships. As a result, an adaptive control of smart city processes can be achieved through context based de-/activation of relevant security and privacy policies.

4) SERVICE DEVELOPMENT TOOLS

The ISCO platform aims at a holistic integration of cross-domain smart city services. Therefore, incorporation of stakeholders in the realization of vertical city applications is critical to achieve the desired impacts on city processes and quality of life. This is especially relevant during the device configuration, service development, provisioning, and integration phases of smart city solutions, which are currently carried out by single manufacturers, platform operators, and solution providers. As a city digitalization layer, ISCO also provides various tool sets for those stakeholders to guarantee interoperable solutions. These tools are referred to as *ISCO City SDK*, which consists of a SCO semantic description tool, service monitoring tool, network monitoring and management tool, and access control policy tool.

Before we describe in detail the three main ISCO components *City Object API* (Section IV), *Service Directory* (Section V), and *Service Orchestration* (Section VI), the next section provides an overarching view of the ISCO workflow for service creation and discovery.

B. ISCO WORKFLOW FOR CROSS-DOMAIN SERVICE PROVISIONING

In the high-level service workflow of ISCO depicted in Fig. 4, service providers can introduce their service descriptions

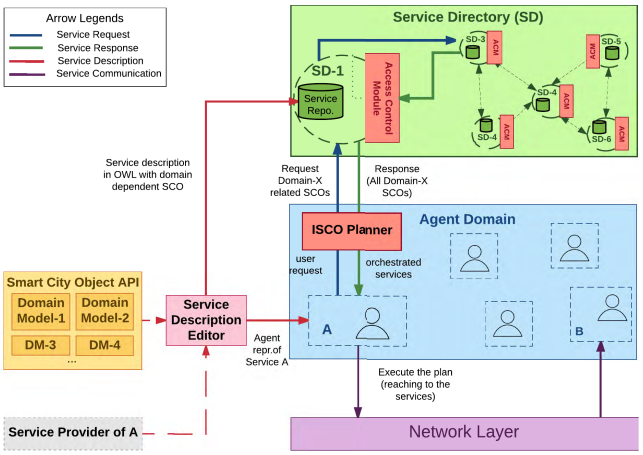

FIGURE 4. Interaction of major ISCO platform components.

using the relevant domain models, after which a software agent representation of the service is automatically created under the ISCO agent domain. The description is also saved within the Service Directory (SD) in the Web Ontology Language (OWL) format similar to [16]. The SD is structured as a dynamic collection of distributed nodes (typically hosted by service providers or other market players throughout the city) and serves as the repositories and request points for registered services. For example, *Service A* in Fig. 4 is registered in the service directory node *SD-1* as the closest SD instance. Every service agent contains an instance of the ISCO Planner to request and orchestrate services. Once a user makes a request, the agent forwards it to find domain-related SCOs from its registered node. Then, the node asks the other nodes in the SD, using an information-centric routing overlay, as described in Section V, eventually returning a set of matching services. The access control module filters out services that are not authorized for the requester. Finally, the domains that contain services of interest with authorized access are returned. The ISCO Planner can now process the services first to filter based on their qualities then to find an applicable orchestration of the services based on the user request. The workflow is further illustrated in Section VII with details about vertical smart city applications that make use of the ISCO platform.

IV. SEMANTIC DESCRIPTION OF SMART CITY OBJECTS

Our fundamental understanding of a smart city is the interconnection of heterogeneous IT-enabled entities, namely physical devices as well as virtual services, that provide a certain functionality: Smart City Objects (SCO). The functionalities provided by the individual entities overlap, are not known a priori, and may even be temporarily unavailable, which complicates and impedes the efficient use of them. Consequently, to enable other entities to utilize such an SCO, an appropriate technical representation is needed. In addition, an API should regulate and facilitate the creation of and access to SCO for applications; something that other

approaches often neglect. In conclusion, a technical representation that takes into account our definition of an SCO and the mentioned problems has to solve the following challenges:

(1) *Functionality*: We define an SCO by the functionality it provides to other SCOs; hence, the representation of such an entity has to reflect this: All information that are needed by a calling entity, such as an orchestrator, has to be accessible.
(2) *Heterogeneous Environment*: The domain of a smart city consists of several sub domains. The model has to give an appropriate way of defining the underlying data model of an SCO so that even a requester from a different domain can utilize it.
(3) *Dynamic Environment*: The technical representation has to be adaptable to a permanently changing environment This means that the data model used by the SCO cannot be statically defined, but also has to be dynamic. This additionally requires a representation that can be semantically understood by the requesting entity.
(4) *Tremendous Amount of available SCOs*: The individual SCOs provide a certain functionality that may be similar or even equal to other ones; in a city-scale environment this can even be thousands of devices or services. The technical representation has to provide suitable mechanisms that allows for further filtering to find the best matching SCO's.

A. DESCRIBING SMART CITY OBJECTS

For a reasonable way of sharing knowledge and modeling SCOs in a heterogeneous and dynamic environment, the usage of ontologies is the de-facto approach [17]; domain and data models described in an ontology provide a sophisticated semantic description that can be parsed by an application. Domain knowledge is separated from operational knowledge and can easily be reused.

1) BACKGROUND

While the notion of an ontology has many different definitions, the most consensus on an ontology is ''being a formal, explicit specification of a shared conceptualization [18].'' Main components of ontologies are the domain-relevant entities, organized as *concepts*, and their (unary and binary) relations, formulated in a logical language. A taxonomy, which defines generalization and specialization relations on the individual concepts forms the basis of the ontology [19]. In the so-called *Semantic Web*, ontologies have become popular due to the possibility to share knowledge that can be parsed by software agents, integrated into their knowledge base to reason about the gathered information and combine different facts to obtain new knowledge [20]. There exist different languages to describe ontologies, which can usually be divided into *Logic Programming (LP)*-based and *Description Logic (DL)*-based languages. Other logic languages such as *Modal Logic*, *Full-First-Order Logic* and *Higher-Order Logic* are more expressive, but do not allow complete or solvable reasoning. *Frame Logic (F-Logic)* [21], as a representative for LP languages, combines conceptual modeling with an object-oriented approach, but is based on the *Closed World Assumption* (that says all knowledge that is true is known as true) and is undecidable, in contrast to, e.g., the *Web Ontology Language (OWL)* [22], the most popular representative for DL languages. OWL is a specification of the World Wide Web Consortium and emerged as a part of the Semantic Web initiative. Another way aside ontologies to describe a domain in a technical form that can later be used for implementations is to use so-called *Domain-Specific Languages (DSL)*; while they share a lot of attributes, they aim at different aspects of the problem domain [23]. There are also DSL approaches for the IoT domain [24], [25], providing the typical DSL benefit of efficient programming in the problem domain, but they are also restricted by the Closed-World Assumption.

A sophisticated approach to model complete sensor (and actuator) environments, especially in the context of the *Web of Things*, with their observations and related properties is the W3C recommended *Semantic Sensor Network (SSN)* ontology [26]. It allows for a complex and detailed modelling, but the provided components focus on the physical devices, are very generic and therefore mostly not usable without further extensions; more abstract components like the notion of a smart city entity are not included. Several approaches that model the *Internet of Things* (IoT) domain in a descriptive way exist. Aligning to ontology-engineering good practices, they usually utilize existing ontologies such as SSN. The IoT-Lite ontology [27], utilized in FIESTA-IoT, has a lightweight approach that gives a good insight on how to efficiently classify the different accessible devices in a smart environment and what basic properties such a model should provide. However, it lacks a functional description that can be called by a requester. An approach similar to IoT-Lite is IoT-O [28], providing an IoT core ontology with focus on modularity and reuse of existent ontologies. Different from IoT-Lite, IoT-O uses the *Minimal Service Model (MSM)*, a lightweight alternative to OWLs, to describe the provided service and therefore allows for an automatic orchestration. Aspects such as context and QoS of an entity, complex devices or services without an attached physical device are not covered by this ontology. Other ontologies for the IoT domain are Smart Appliance REFerence (SAREF) [29], iot-ontology [30], Spitfire [31], IoT-S [32]. Common to all of these ontologies is the concept of sensing and actuating devices that are accessed using a service.

In describing the SCOs in terms of their functionality, it seems best to see them as (web) services. The service-oriented computing community developed well-established standards to describe and invoke functionalities, such as WSDL/SOAP and REST services, but these standards do not suffice a smart city. Semantic descriptions of software services provide the needed additional information layer, for which several approaches have been proposed in recent years. Amongst others, these are OWL-S [33], WSMO [34] and SAWSDL [35], which extend typical service

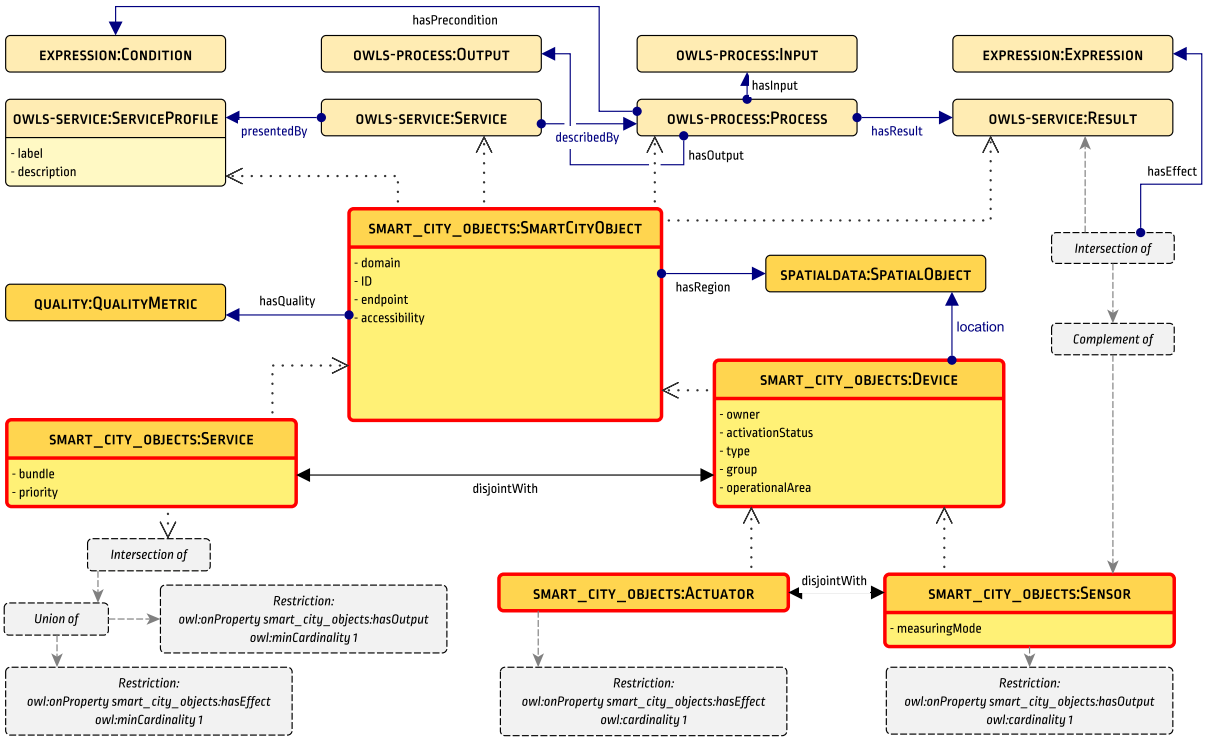

FIGURE 5. Centralized overview of the ISCO Smart City Object ontology.

information by *preconditions (P)* and *effects (E)* (so-called logical specification), and define *input (I)* and *output (O)* (so-called logical signature) information in a more expressive ontology language, such as OWL [36]. However, these semantic approaches only address web services and not the smart city domain in particular. Therefore, in our approach we extend the OWL-S ontology for smart cities, since it allows for describing *IOPE*-based (full-functional) services, which is needed for sophisticated automated service orchestration. Other approaches such as the MSM, which only allow for describing the logical signature, are not expressive enough. Nambi *et al.* [37] describe a semantic knowledge base for IoT and use the OWL-S ontology to describe services, similar to our approach. However, the authors focus on describing the domain holistically with contextual information, and therefore create a rather complex multi-layered ontology; furthermore, orchestration of services is neglected in the architecture presented in [37].

Within a city-scale environment, certain functionalities are provided by multiple entities; for these functionally similar (or even equivalent) entities, non-functional attributes can be used to further narrow down to the best SCOs, i.e., quality of service (QoS) parameters. With many existing approaches for QoS ontologies, Kritikos *et al.* [38] as well as Tran and Tsuji [39] give good overviews over the research topic. Basically, Kritikos et al. have identified three layers to define a QoS ontology: 1) the Service Quality Meta-Models (SQMM) define the concepts needed to instantiate the 2) Service Quality Model (SQM), which describes concrete quality properties. 3) Lastly, Quality-Based Service Descriptions combine the SQM with values and constraints to allow for matchmaking services. The onQos ontology [40] is such a three-layer ontology, which extends the OWL-S ontology. onQos is used for specifying advertisements of the service providers as well as requirements of the service requester. It is a well-structured approach with only a rudimentary concept for the quality scales.

We use [41] with the extension of graded memberships [42]. Our SQMM can be seen in Fig. 6. An example for a SQM is shown in List. 1. The processing of these criteria will be addressed in Sec. VI.

2) ISCO APPROACH

Our extension is inspired by IoT-Lite and IoT-O with a focus on the orchestration; it integrates devices and further extends the concept of a service by attributes needed by an SCO, e.g., the location or the accessibility. An overview of our ontology can be seen in Fig. 5, where the central class is *SmartCityObject* that is a subclass of the OWL-S classes *Service*, *ServiceProfile*, *Process* and *Result*. This means that in terms of OWL-S, SCOs represent both the service as well as the underlying process with its result and, on the other hand, guarantees that SCOs can be used by systems that understand OWL-S services.

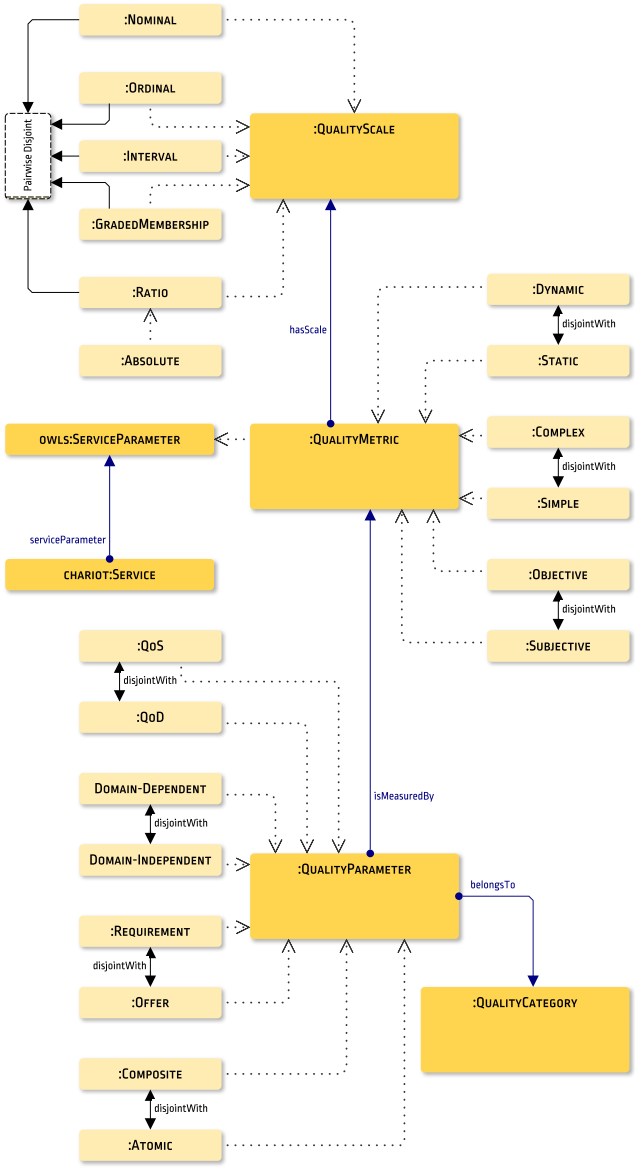

FIGURE 6. Our quality-of-service ontology.

Besides virtual or digital services, a significant part of a smart city is represented by devices. Since we model the domain in terms of their functionality, we distinguish here between *Actuators* and *Sensors* that only differ in terms of what they provide: An actuator changes the world's state, thus has an effect, while a sensor measures a kind of quantity, thus only provides some output. By this definition we are able to describe smart city devices in the same way as virtual services.

The large amount of potential SCOs with similar or identical functionalities in a smart city pose a challenge on service discovery and planning operations. Here the design of the semantic model plays a crucial role, as the amount of

```
Prefix: qos: <http://.../QualityOfService.owl#>
Prefix: rdfs: <http://www.w3.org/2000/01/rdf-schema#>


Class: Authorization

  Annotations:
    rdfs:label "Authorization"@en,
    rdfs:comment "Only authorized principals can access
         the service."@en

  SubClassOf:
    qos:AtomicQualityParameter,
    qos:Domain-IndependentQualityParameter,
    qos:QualityOfService,
    qos:QualityParameter,
    qos:belongsTo value Security

  Facts:
    qos:isMeasuredBy  AuthorizationMetric


Class: AuthorizationMetric

  EquivalentTo:
    qos:hasScale value authorization_scale

  SubClassOf:
    qos:ObjectiveMetric,
    qos:QualityMetric,
    qos:SimpleMetric,
    qos:StaticMetric

  Facts:
    qos:isQualityMetricFor  Authorization


Individual: Security

  Types:
    qos:QualityCategory


Individual: authorization_level_authority

  Types:
    qos:Classificator


Individual: authorization_level_citizen

  Types:
    qos:Classificator


Individual: authorization_scale

  Types:
    qos:NominalScale

  Facts:
    qos:hasClassifier  authorization_level_authority,
    qos:hasClassifier  authorization_level_citizen


Individual: example_service

  ...

  Facts:
    qos:qualityParameter  authorization_metric,


Individual: authorization_metric

  Types:
  AuthorizationMetric

  Facts:
    qos:hasValue  authorization_level_citizen
```

Listing 1. Example of a QoS parameter used in a service *example_service* (in Manchester Syntax)

returned SCOs can be reduced drastically with the possibility for a fine-grained description of the required service. In this connection, our model allows assigning SCOs to (multiple) sub domains of the universal smart city domain, whereby we refer to the overview made by Neirotti *et al.* [43] for the individual domains. Furthermore, devices can refer to a type in a taxonomy to better describe their nature. As another important feature, non-functional attributes that facilitate a better selection of SCOs are related to the geospatial information of the entities, i.e., the regions for which an SCO provides its functionality as well as the location of physical devices, whenever applicable. Lastly, management attributes such as the provider of the SCO, the type of accessibility to provide security, attributes for grouping SCOs and unique identifiers are adopted to ensure a working and efficient autonomous utilization of such entities.

B. CREATING AND ACCESSING SMART CITY OBJECTS

An integrated API facilitates the usage of SCOs by city applications and stakeholders. Because of the special requirements of a smart city, especially the huge number of entities, and the semantical data format the SCOs are stored in a distributed triplestore. Entities can be queried using SPARQL, however, this approach is cumbersome. The API provides adequate methods for easy access to the city objects: SCOs can be queried directly by their ID, or a more generic search can be triggered based on attributes such as the domain, input and output parameters, quality measures, and overall score, after which the matching SCOs are returned. The API also converts between the ontology representation used in the service directory and an object-oriented class model that can be used directly by application developers. This allows application developers to implement ISCO-compatible software by just employing the API. Conversely, service developers can generate the service description object-oriented, add information on how to access the SCO using the *endpoint* attribute and then use the API to register and upload the resulting SCO to the service directory.

1) BACKGROUND

A key challenge when implementing an API is how to give (programmatic) access to the ontology. Basically, two approaches exist: 1) convert the ontology into a direct software model, either by hand or using tools, or 2) use libraries that allow working with the ontologies, leading to an indirect model. A third option, a so-called hybrid model [44], uses the first option for core classes and the second for domain-related classes. Direct models can be retrieved using so-called *Model Transformation Languages*, such as ATL,[11] using annotations to connect object-oriented and ontology classes [45] or use custom transformations [46], [47]. However, most of the available implementations, if still working at all, only provide a very generic and basic transformation that needs much manual work later on. The most established solutions for indirect models are *OWL API* [48] and *Jena API*.[12]

2) ISCO APPROACH

For our approach, which picks up the hybrid model, we use EMF[13] to generate a class model of the presented ontology that can be used by application developers. We assume here that the developers, utilizing the SCO API, will take care of how to integrate domain-related objects, possibly using an indirect model. Since the service directory is implemented using the Jena[14] project, a converter class converts between the class model and a Jena model representation. Jena then offers methods to transform this Jena model to an ontological representation, which can be used by a SPARQL connector to exchange information with the service directory.

The SCO API is also integrated into Semantic Service Manager [49], a graphical user interface for creating SCO descriptions, and Visual Service Design Tool [50], a tool for defining agent behaviour based on Business Process Modeling Notation (BPMN); this agent then realizes the actual process defined by the SCO description. These two together function as a service developer tool that facilitate the creation of SCOs.

In summary, the SCO API functions as a bridge between the other components in the ISCO architecture to allow for easy implementation of ISCO-compatible applications that build on the functionalities provided by smart city objects.

V. SCALABLE SERVICE LOOKUP AND DISCOVERY

In this section, we discuss the challenges of managing smart city information and propose a design for city-scale distributed directory of smart city objects, which also facilitates dynamic end-to-end service composition.

A. MANAGING CONNECTED OBJECTS

Smart city systems have to deal with billions of devices and services, which are often combined together in order to provide city applications. Such devices and services may need to be uniquely identifiable for tracking or establishing connections, for which the *naming* and *addressing* mechanisms are designed. The former is concerned with creating a labeling scheme or attribute that differentiates an object in a local or global scope, while the latter is concerned mainly with locating the object or service and enabling their interactions. Most of the existing IoT or smart city solutions are built on the currently well established architectures and data communication technologies such as Service Oriented Architectures (SOA), the Internet Protocol (IP), and many others. While those technologies have well-defined mechanisms to address and locate devices and services, SCOs pose new challenges to their management, ascribed to the smart city service requirements: i) heterogeneity of devices and

[11] http://www.eclipse.org/atl/

[12] https://jena.apache.org/documentation/ontology/

[13] https://www.eclipse.org/modeling/emf/

[14] https://jena.apache.org/documentation/tdb/index.html

services, ii) complex human-service interactions and behaviors, iii) cross-domain integration, and iv) highly dynamic and mobile environment.

In the following, a brief overview of the identification and discovery approaches in the IoT domain is provided, motivating our solution concept for the distributed ISCO service directory.

1) OBJECT IDENTIFICATION

In most use-cases, for products and physical objects, standardized naming schemes are widely applied globally in projects SmartAgrifood, CEN TC225, i.e., Barcode, RFID/AutoID, DOI (by International Organization for Standardization), and Electronic Product Code (EPC) (by EPCGlobal). They are referred to as legacy naming and extended for IoT systems in projects SmartAgrifood, CEN TC225. Other naming schemes have specific identifications based on, e.g., URI, IPv4, IPv6, hash value, which do not directly refer to the physical objects. They allow organizing physical domain objects with notations of groups, sub-groups, and abstract entities for cross-domain object identification. Despite the need for a specialised entity management component, those schemes allow more flexible and cross-domain object identification required by complex IoT systems. Thus, they are used in various projects, e.g, IoT-A,[15] IoT6,[16] OpenIoT,[17] for cross-domain object identification.

The choice of addressing schemes are influenced by data transport mechanism and discovery methods. Networked devices may take advantage of IP address and distributed Internet routing as in IoT-A and IoT6 projects. For non-IP devices, a centralized infrastructure is required, which provides mapping service between the entities' names and access methods, e.g., endpoint URI, local gateway. Such infrastructure is designed in most aforementioned projects, which employ distributed DNS, Web service, or Semantic/Linked Data backed directories. Naming and addressing have a close relationship, which affects the design of scalable object discovery.

2) DISCOVERY ARCHITECTURES

Due to the heterogeneity and cross domain requirements, smart city IoT solutions rely on an object discovery infrastructure to provide descriptions about their attributes, location, and access methods, among others. Depending on the application, discovery can be a stand-alone service or integrated with the entity management and gateway functions of an IoT middleware. Nevertheless, it aims at providing scalable services for object registration, mapping, and lookup. As such, object descriptions can be distributed based on logical domain, geographical location, or platform specific hierarchy:

- *Domain-specific discovery* builds on the Domain Name System (DNS) of the Internet and is adopted by some projects such as IoT6 by leveraging IPv6 and proposing service discovery (DNS-SD) [51] and mDNS [52] discovery protocols. Global object clusters are discovered with DNS-SD based on IPv6 and higher level protocols CoAP [53]. In local groups, the multicast mDNS is employed for automatic registration and discovery of devices. The Object Name Service (ONS) [54] used in SmartAgriFood is a similar service to DNS for discovery and resolve physical object with EPC code. A local ONS server looks up product descriptions for scanned code by mapping it to a set of resource descriptions provided by external services. SmartAgriFood employs ONS for discovery of entities in production chains in conjunction with a decentralized semantic storage for object data.
- *Geolocation based discovery* is common in device centric and location based applications. The objects are addressed based on their notation of geographic points, areas, or network cluster. While the indexing and geo-discovery are straightforward, additional resolution infrastructure is still required to provide operational details of the resources, as in, e.g., IoT-A. BUTLER projects suggests a two-level hierarchy of catalogue and resolution servers. Catalogue servers first resolves resolution servers according to the areas they covered. The resolution servers store the actual resource descriptions.
- *Service Directory (SD)* A directory structure holding rich descriptions of IoT entities is often required in addition to previous distribution approaches. Beside accessibility description, attributes about the entities and their relationships provide data needed for, among others, management, service composition logic of the applications. Semantic web approach is adopted by the projects and is referred as semantic discovery. OWL-based ontologies capture models of physical, logical entities and their relationships, e.g., device capabilities, clustering, QoS requirements. The semantic descriptions allow discovery and matching of services or devices at runtime using SPARQL queries. Additionally, new facts about objects can be inferred through logic reasoning, which adds intelligence to the application compared with other statically captured models. To meet the required scalability, directory services consist of distributed servers (nodes), which are organized as a multi-level hierarchy or peer-to-peer topology. In contrast, we propose a scalable service directory infrastructure, which features a flat, self-organized topology of distributed service directory nodes.

B. ISCO SERVICE DIRECTORY BASED ON AN ICN OVERLAY

The IoT service directory (SD) in ISCO is designed to self-organize the distributed storage and retrieval of smart object descriptions. Its flat architecture makes the directory eligible for universal service discovery for IoT by removing the

[15] http://iot-a.eu/
[16] https://iot6.eu/
[17] http://openiot.eu/

dependency on discovery mechanism from specific applications and domains. Before presenting our SD design in ISCO, we first provide a short introduction to ICN as an important enabler for our design.

Information-Centric Networking: The underlying principle of ICN is that a communication network should allow a content consumer to focus on the data it needs, named content, rather than having to reference a specific, physical location where that data is to be retrieved from, i.e., named hosts, as in current Internet architecture. ICN offers a wide range of benefits, e.g, content caching to reduce congestion and improve delivery speed, simpler configuration of network devices, and building security into the network at the data level. Communication in ICN is driven by data consumers, through the exchange of Interest (INT) and Data (DATA) packages. Both types of packets carry a name that identifies a piece of data. The consumer sends an INT with the name of the data it needs. When an intermediate node receives the INT, it looks for the data in its content store (CS). If the data is not found, it forwards the INT to the next nodes and keeps track of the incoming and outgoing network interfaces for this data in pending interest table (PIT). A series of such forwarding actions creates a breadcrumb path the INT has passed. When the INT arrives at the source node, the requested data is put into DATA and sent back the path towards the consumer. A previous forwarding node receives the DATA, it removes the data name entry in PIT table and adds an entry with the name and network interfaces to forward the data to consumers in the FIT table. Intermediary nodes on the path cache the DATA in their CSs for subsequent INTs. ICN offers a wide range of benefits, e.g, content caching, simpler configuration of network devices, and security at the data level.

We apply ICN's data centric paradigm, more specifically the *Named Data Networking*[18] as a realization of the ICN approach, for the distribution of object descriptions among SD-Nodes in our service directory design. The solution can take advantage of the aforementioned ICN features for IoT requirements due to the data-centric nature of many smart city and IoT applications. Based on those features, refined mechanisms are designed for SD functionalities, i.e., developing attribute-based object query methods and content caching strategies for reduced storage overhead as well as increased responsiveness and accuracy.

1) ICN BASED NAMING SCHEME FOR CITY OBJECTS

In the service directory infrastructure, the data to be exchanged are descriptions of services and devices, which mainly contain various attributes. Using ICN enables the SD to decouple the data from locations of the nodes that store the data while taking advantage of ICN forwarding and caching mechanisms. Each SD-Node acts as an ICN router, which serves the requests for object's description by its name, or forwards the requests towards other nodes holding the description. Therefore, the design of a naming scheme

[18]https://named-data.net/

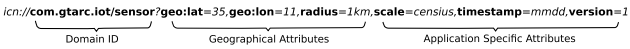

FIGURE 7. IoT resource naming scheme in ICN based Service Directory.

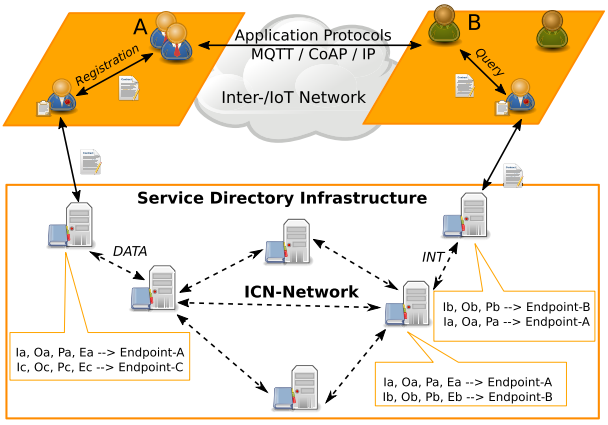

FIGURE 8. ICN-based Service Directory.

affects the performance of object discovery. ICN naming adopts the semantics of Universal Resource Identifier (URI) scheme. However, the host part does not imply location of the resource, but rather identifies its owner or search domain. The *attributes* part enables the expression of resource attributes that can be used to look up and discover the resources regardless of where they are stored. An example ICN name is shown in Fig. 7, which contains rich semantics describing a sensor's domain, location, type, etc.

Matching of query attributes and the semantic descriptions is handled by a matcher component in each SD-Node. For this purpose, attribute names can refer to their semantic description by using popular ontology prefixes, e.g., the *geo* namespace in the example. Depending on the use-case, a strategy to store descriptions and to forward the requests based on object attributes can be dynamically configured. Additionally, various caching and forwarding strategies can be designed to best serve the query demands and SD infrastructure performance.

2) SERVICE DIRECTORY NODE ARCHITECTURE

The ISCO service directory is constituted by a distributed collection of SD nodes as depicted in Fig. 8. The design of an individual SD-Node, which contains semantic descriptions of SCOs, is shown in Fig. 9. We employ a triple store for the storage and query of SCO attributes. Various interfaces are implemented for respective transport protocols for the query and distribution of the descriptions. Each functionality is implemented as a modular component based on the OSGI platform architecture. Descriptions of the most important components are provided next.

Triple Store (TDB) is a component of the Jena project, which serves as a high performance RDF store for the directory server. It provides an API as well as a SPARQL interface for storage and query of semantic descriptions.

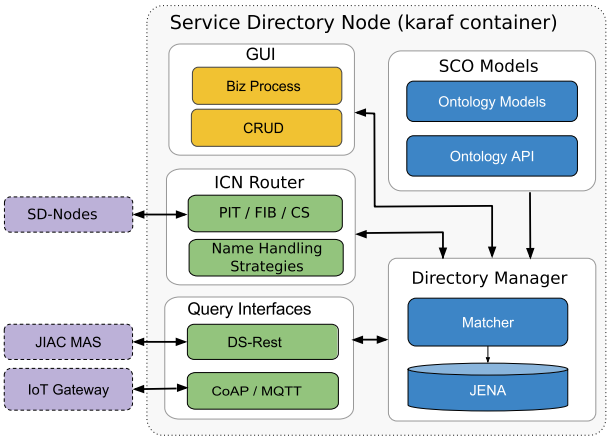

FIGURE 9. SD-Node Architecture.

Matcher component implements mapping methods between requested search attributes and suitable SCO descriptions, which are potential search results. It handles queries from IoT service components and other SD-Nodes received through different query interfaces.

ICN Router enables connectivity between SD-Nodes with ICN transport protocol to form a distributed SD infrastructure. Discovery of SCO descriptions is realized by the exchange of interest messages with SCO names. The available CCN-Lite [55] solution is extended with the implementation of strategies for forwarding requests and caching SCO descriptions among SD-Nodes.

Query Interfaces provide distributed application protocols, which allow higher level services to access SCO descriptions and ontologies. It employs various application transport protocols, such as CoAP, MQTT and Rest. To enable semantic expressions and functional description in the queries, the RESTdesc [56] approach is applied. It allows certain matching rules to be embedded, which improves the accuracy of SCO matching and discovery.

C. SCALABILITY OF ICN BASED DISCOVERY

An ICN architecture, in contrast to a host-centric one, does not dictate a predefined hierarchy, e.g., conformance with IP routing or a specific discovery protocol (DNS), among others. This results in a flat network with self-organized topologies. The attribute-based discovery only depends on how an approach describes the devices and services, specifically, their semantic models, matching approach, and strategies for information organization. Fig. 8 illustrates a distributed SD infrastructure utilized by the ISCO platform based on a multi-agent system architecture. The agents on the ISCO platform are logical representations of SCOs. Once a service or device is made available, the agent (A) registers its service in the SD by sending SCO descriptions to a nearby SD-Node, making it the source of the SCO description. Due to the flatness of the ICN transport network, no constraint is given on the placement of the SD-Nodes. Globally dedicated nodes or, if required, managed nodes for each local domain or organization could be used. If a service (B) wants to look for the description of another service (e.g., A), it sends the request to a local SD-Node in an ICN interest message. The request is forwarded to the source SD-Node of the description, which results in replications of the description in the individual caches of SD-Nodes along the query path. Forwarding and caching strategies can be adapted; e.g., the choice of lifetime of the replicas implies a trade-off between the dissemination of descriptions closer to requesting agents, and timeliness, consistency of the information. Some caching approaches are discussed in [57]. Moreover, the self-organizing topology allows additional SD-nodes to be added or removed as required by deployment scenarios, querying patterns, among others. Applying cloud computing or container technologies (e.g., Docker) enables elastic provisioning of SD services.

VI. SERVICE COMPOSITION AND PLANNING

Web service composition (WSC) has been widely applied [58] to create new value-added services from existing atomic ones. The QoSs of all services involved in this composition affect the quality of the composite service. In this section we present the *ISCO middleware planning layer* which applies WSC to IoT components. Planning in IoT environments is a challenging task: IoT instances may appear, disappear or move, thus, the promised Service Level Agreement (SLA) can change over time. Applying WSC in such a fragile heterogeneous and dynamic environment requires an adaptive and self-optimizing architecture. Therefore, we adopt an extended version of the MAPE-K [59] architecture, which enhances the system with significant adaptation capabilities.

A. BACKGROUND

Implementing new services for smart cities is a complex task, as service developers have to identify and bind the required set of services and sensors with the highest quality of service and data out of millions of possible SCOs at design time. The implemented services have a static binding to dynamic SCO instances and have therefore low adaptation capabilities and stability. Maintaining such services is also costly. The service developers have to monitor the system dependencies to detect qualitative or functional changes. Those changes require a code revision for the sake of replacing misbehaving or outdated SCOs. The ISCO middleware eases the implementation and deployment of such complex services by abstracting the service and device layer allowing service developers to focus on the functional and qualitative definition of their services.

There is a large research literature on web services composition, which forms the basis of our planning solution for SCOs. Rather than providing an incomplete list of such related work here, we point to a representative set of surveys on the topic, [60]–[64], through which interested readers may get a more comprehensive overview. In the

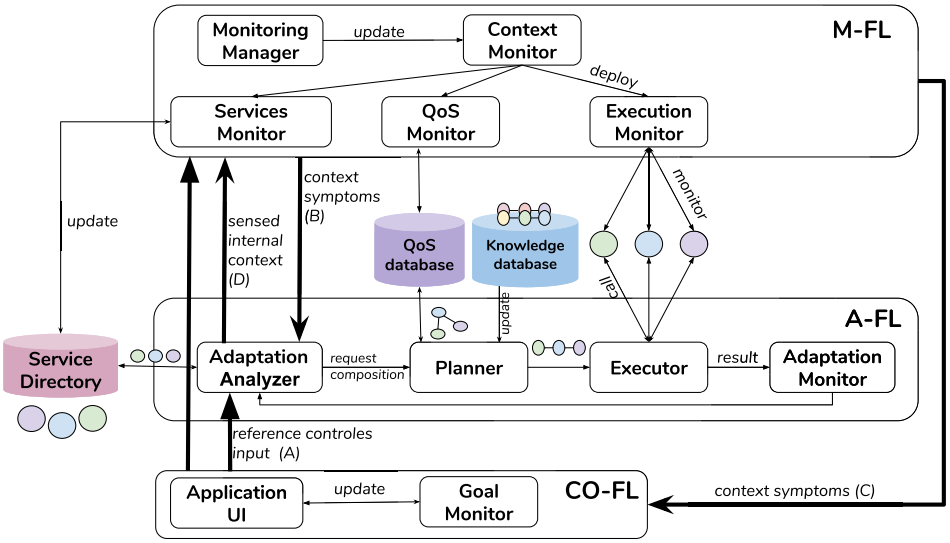

FIGURE 10. An overview of the ISCO planning layer components.

surveys [60] and [63], the authors rely on the composition life cycle (define, select and execute) to compare existing approaches. The meta-survey in [61] reviews available surveys and classifies them in four main categories: service combination centric surveys, service selection centric surveys, combination and selection-hybrid surveys and service discovery centric surveys. Pejman *et al.* present in [62] several composition methodologies and group them in two main clusters: methods using evolutionary algorithms (such as Genetic Algorithm, Immune System algorithms and ant colony Optimization) and methods that are not using evolutionary algorithms (such as Graph based planners). A review of the current QoS-aware web service composition approaches and a classification of the supported QoS properties is given in [64].

B. ISCO PLANNING LAYER

The ISCO middleware planning layer is responsible for generating a service composition, involving different SCOs, that satisfies the specified – functional and qualitative – application requirements at runtime. The adaptability of this module is crucial, as it has to suit different contexts and serves a wide range of applications with varying goals. We are therefore applying the DYNAMICO [65] design guidelines for adaptive systems to enhance the planning layer adaptiveness. This reference model defines three levels of dynamics: the *Objective Feedback Loop (CO-FL)*, which is responsible for managing changes in control objectives, the *Adaption Feedback Loop (A-FL)*, which models the adaptation mechanisms, and the *Monitoring Feedback Loop (M-FL)*, which tracks context and objective changes. The ISCO planning layer architecture is depicted in Fig. 10 based on this three-level arrangement. Each component of the architecture is detailed next.

1) OBJECTIVE FEEDBACK LOOP

In dynamic software systems, the adaptation goals (or objectives) should be defined and monitored. These control objectives can define functional system requirements or refer to non-functional system properties (e.g., QoS) [66]. The monitoring of these adaptation goals needs an explicit formalization, which e.g., is accomplished in AI planning through a goal state. The goal state contains the facts the system should achieve. In ISCO, we are using an IOPE (input, output, precondition and effect) representation to define the functional goals (e.g., plan a trip) and QoS to state the non-functional system requirements (e.g., trip cost and duration). During the execution of software systems in dynamic environments, the adaptation goals might be affected by several changes, including, but not limited to:

- *Goal change:* Because of the changes in the environment, a goal might no longer be valid. Users can also explicitly modify the adaption goals.
- *Goal no longer reachable:* During the execution of a plan, the environment might change, e.g., through disappearing services or changes in the condition of the environment. Those changes might lead to situations where the goal is no longer reachable, given the current plan.
- *Goal order change:* While executing a plan to reach a certain goal, the priority of goals might change, e.g., if the QoS of used services changes. With a change in QoS the goal might become less attractive, and with that, a new goal could be perused. By changing priorities, a certain plan might lose effectiveness or efficiency.

An adapting system, especially in dynamic environments like the IoT, should be able to monitor its goals and evaluate when to start switching objectives. Switching a goal might cause the abortion, adaption or recreation of a plan, e.g., if weather

change influences a part of a journey, and riding a bike is no longer a valid option. In the DYNAMICO Framework [65] the CO-FL addresses such problems.

2) ADAPTATION FEEDBACK LOOP

(A-FL) receives the adaptation goals from the CO-FL and the monitored context information from the M-FL and selects the appropriate adaptation mechanism to maintain or reach the specified goals. Our A-FL layer implements different approaches that enable the system to adapt to system-wide changes. In this loop, the adaptation process might be initiated due to changing control objectives or context information. The A-FL has four main components introduced below:

Planner is responsible for combining SCOs by connecting their IOPEs for the sake of generating new composite services that satisfy the client application requirements. Our approach is extending the traditional QoS-aware WSC to support IoT devices and sensors. The generated plans should fulfill the functional and qualitative system requirements defined by the CO-LP. The current implementation is graph based, and uses a Fast Forward planner [67] to reduce the search space and fasten the system response. The search graph is composed of four different types of nodes representing the different entities of the system: Input/Output, Precondition/Effect, Service and Goal/Initial state nodes. Each node type has its own activation function that defines the state of the node. The service nodes for example are only active if all precondition and input nodes are active. Active services might generate output and will activate the connected effect and output nodes. Due to the extremely large number of IoT-enabled devices and the software representation of their functionalities, it is fundamentally necessary to be able to fall back on efficient heuristics in order to help the overall system achieve better performance. To reduce the search space and fasten the planning process, we use the functional service description and filter out all nodes that are not reachable from the initial state node (all services that cannot be executed) using fast forward graph search as well as all nodes that are not relevant for the current request by applying a fast backward search. The distance from the initial state and to the goal state as well as the current path length and number of executions are some of the heuristics that have been tested.

Interpreter is responsible for the execution of the composite services. The execution is monitored whereas the current state is forwarded to the analyzer module. This process may fail or the results may deviate from the specified goals. In this case, the adaptation analyzer should trace those deviations and replace the missing or misbehaving components to maintain the system robustness. The analyzer might also stop the execution process if needed, e.g., if the adaptation goals were updated.

Adaptation Analyzer is the central component of the A-LP. This module evaluates the current adaptation goals, selects the most suitable adaptation mechanism and initiates the adaptation process. It also identifies and deploys the required monitoring modules. This module stores – for each new request – the generated service composition along with its global QoS in the knowledge database. If the system goals or context are updated, the analyzer will first search the knowledge database for a service composition that meets the current system requirements. The freshness of the solution, if found, is controlled. If the solution does not exist or is outdated, the planning process is initiated. The generated plan is then forwarded to the interpreter; finally, if the execution is successful it is stored in the Knowledge database. If the execution fails, the planning process is re-triggered in order to replace the unavailable service(s).

Adaptation Monitor checks the state of the adaption mechanism. The adaption needs to change if the adaption mechanism itself is no longer adequate for the system. This monitoring is done to observe the performance of the adaption mechanisms in case the adaption mechanism itself needs to be adapted. This inadequacy could be the case, e.g., if the goals or the context change faster than the adaption can react. In this case, the adaption mechanism might neglect an optimal solution to speed up the adaption. This adaption mechanism is modeled in the planning component. Depending on the heuristic used, the planning can adapt to goal changes, e.g., if different quality parameters become important, or by adapting the planning parameters like relaxing the optimality of A* to a ϵ-admissible heuristic [68] to speed up the search for a solution.

3) MONITORING FEEDBACK LOOP

Self-adaptive systems need to maintain their context-awareness relevance, in order to adapt at runtime to changing context. For the sake of preserving the system context-awareness, different monitoring strategies might be applied depending on the current adaptation goals. The M-FL is the context manager in the DYNAMICO reference model (see Fig. 10). The M-FL deploys different context gathers, which monitors the current system context, and reports updates to the A-FL. Our ISCO platform implements four different monitoring components each of which is targeting a specific system component or process:

QoS Monitor is responsible for monitoring the QoS parameters of the supported services and devices. The measured QoS at runtime may deviate from the defined SLA used to generate a service composition, and should therefore be updated. To guarantee optimality, the system has to adapt to these changes. By changing QoS, the service composition has to be adjusted and the monitored QoS has to be taken into account during the replanning process. The current version of the QoS monitor measures network-specific global parameters, relying on the JIAC agents technology. This module is

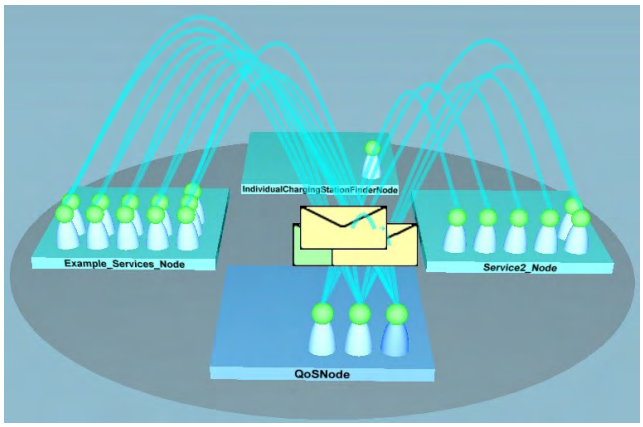

FIGURE 11. The QoS monitoring component.

composed of several JIAC agents (see Fig. 11). Those agents are able to discover and communicate with all service agents running within the ISCO platform. Each monitoring agent monitors specific QoS parameters for a defined subset of services. The number of running agents and services is monitored by the monitoring manager, which is able to start (or kill) monitoring agents if needed. This is for example the case, when new services are created. In this case, the service-to-agent distribution is analysed and the new services are assigned to the running monitoring agents. The monitoring manager is the heart of this self-healing and self-monitoring module. Its availability is monitored by the monitoring agents, that are able to replace it or restart it if needed.

Services Monitor – As mentioned before, service developers are able to create new services or update the functional requirements of their services. These updates have to be considered during the planning phase in order to guarantee the optimality of the final composition. This component observes the service directory and notifies changes to the adaptation analyzer.

Execution Monitor – During the execution of a composite service several issues may arise (e.g., timeout exception, network exception, the returned values does not have the right format). This module reports the tracked issues to the adaptation analyzer, which then initiates the most appropriate recovery process, e.g., replacing unavailable services with similar ones or generating a new sub compositions.

Context Monitor captures changes in the context of the adaption system. This monitoring is done to be able to adapt to changing conditions in the environment, e.g., changing legal rules of the planning domain.

VII. ISCO USE CASES

We present two representative use cases that were implemented in the ISCO project to illustrate how different components interact with each other and jointly enrich service creation and consumption for various stakeholders in smart cities. The first scenario focuses on a mobility assistance application, mainly addressing the end user perspective in addition to the service provider perspective. The second scenario considers a city surveillance use case during emergency response or event management operations, mainly addressing the perspective of city authorities.

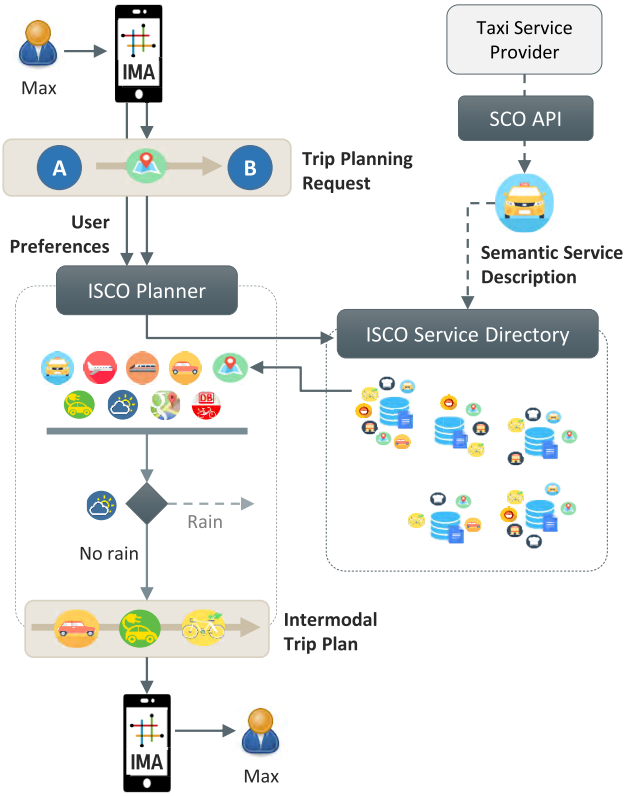

FIGURE 12. Flow diagram and ISCO components' involvement for the intermodal trip planning use case.

The details of each use case is presented in the following sections, while Table 2 provides a compact view of how ISCO framework can assist different types of stakeholders in the city. Furthermore, the overall flow of each use case is illustrated in Fig. 12 and Fig. 15, also highlighting the involvement of ISCO framework components.

A. INTER-DOMAIN INTERMODAL TRIP PLANNING

In the storyline of the scenario, there is a big concert event in Berlin. Max, planning to attend the concert, lives far away from the event venue. He already has the ISCO-compatible *Intermodal Mobility Assistance* (IMA) [69] application installed on his mobile phone. Here, *ISCO-compatible* refers to the fact that IMA app implements proper interfaces to the ISCO service directory and ISCO planning components to discover and access the smart city objects.

Through the SCO API, IMA app integrates various services related to the mobility domain, such as public transport service, car/bike sharing service, weather service, electric vehicle charging service and car parking service, for computing the optimal travel options in a given context. As depicted in Fig. 12, each of these services are registered

TABLE 2. Summary of how ISCO framework components are exploited by various stakeholders in both use cases.

Use Cases	Stakeholder			
	City Authority	SCO Developer	Service Provider	End User
Inter-domain Intermodal Trip Planning	–	Utilizes *SCO API* and *ISCO tools* to semantically describe the functionality and features of its device or service, ready to be consumed by other mobility services or apps.	Deploys its mobility service on *ISCO Service Directory* and benefits from the city-wide distribution channel for offering its service.	Receives customized intermodal route plans through the *ISCO-compatible app* IMA, which builds on the generic *ISCO Planner* for service discovery and chaining.
Dynamic Video Surveillance for Public Safety	Gains streamlined access to both public and private cameras to track a suspect in an area of interest within the city, using the *ISCO Service Directory* and *Planning* components.	Integrates IP camera into the *ISCO Service Directory* with certain access controls with the use of *SCO API*.	Provides a video analytics service that accepts authorized video feeds for real-time processing and annotation, utilizing the *SCO API* and *ISCO Point-to-Point Protocol*.	–

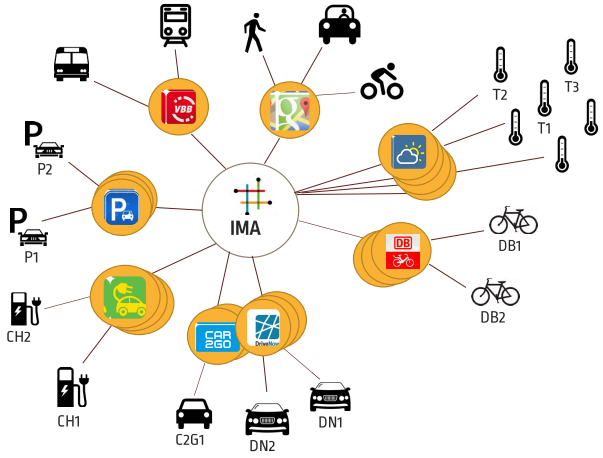

FIGURE 13. Intermodal Mobility Assistance (IMA) app using the ISCO framework to discover and orchestrate various mobility-related services in the city: local public transport service by VBB, bike sharing service from Deutsche Bahn (DB), car sharing services from Car2Go and DriveNow (C2G and DN, respectively), car parking & charging service (CH), regular car parking service (P), mapping service by Google, and weather & temperature sensor service (T).

FIGURE 14. Intermodal route planning results in IMA App, presenting different route options with their time, cost and carbon emission features.

with ISCO platform by service providers using the ISCO API and tools. In our scenario, we also assume an identity and payment solutions provider that has its service integrated on the ISCO platform, decoupling such business enablers from the actual services themselves, each of which can be provided by different entities. ISCO planner can then seamlessly chain mobility services from different providers, without requiring the users to explicitly maintain individual subscriptions to each mobility service.

Max runs the IMA app to plan a trip from his flat to the concert area. IMA initiates a mobility service request, which incorporates Max's preferences, his digital identities, subscribed payment services, and other details relevant for service composition. ISCO planner prepares and forwards the request for the transportation SCO domain model and the service directory returns the relevant services with Max's access rights. Fig. 13 illustrates all available services Max has access to in our use case implementation. These services are then filtered by the ISCO planner through the quality criteria defined by Max in the app, such as cost, response time, location of interest, as well as the service agreements defined by the service providers and security policies defined by Max's identity providers. As a result, matching public transport (VBB), bike sharing (DB), car sharing (Car2Go) services are selected along with the weather and car parking with charging station services (CH) based on the locations of the trip, and Max's service use is monitored and made available for the selected accounting and payment services.

ISCO Planner instance in the IMA app then calculates suitable route options, each presented to the user with their total cost, time and environmental friendliness, as depicted in Fig. 14. The app can also make suggestions based on the

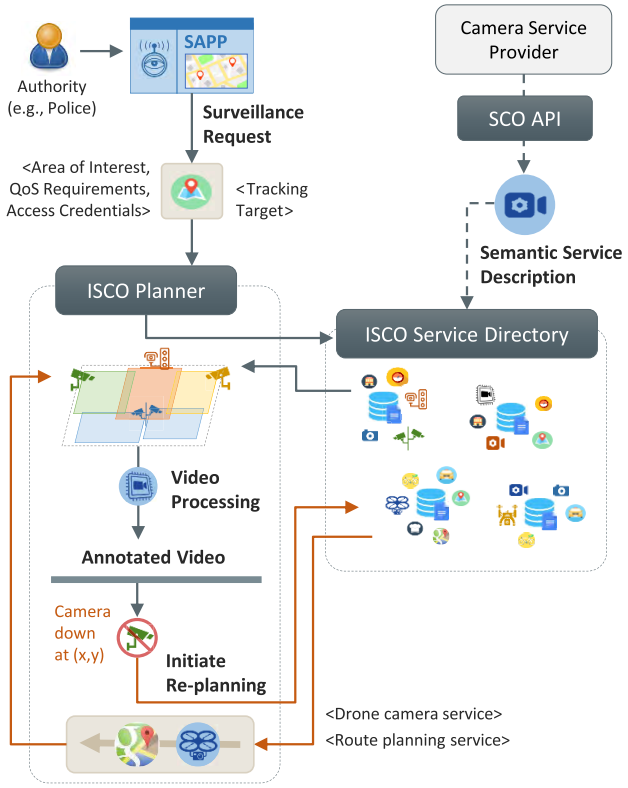

FIGURE 15. Flow diagram and ISCO components' involvement for the dynamic video surveillance use case.

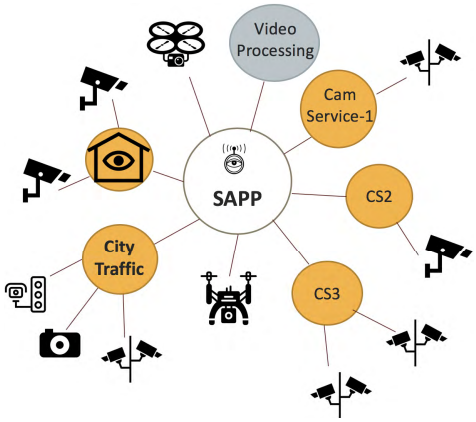

FIGURE 16. Surveillance App (SAPP), integrating various camera streaming services (CS) and video processing services for critical object detection.

weather report, e.g., it may note that the use of bike sharing is not recommended due to rainy weather. Max selects one journey plan and IMA executes the selected services with the order in the plan. In our example scenario, IMA suggests Max to get his electric car, drive to the closest charging station around the concert area (taking into account the status of the car battery), park the car there in CH1 (i.e., a car park with charging station, as illustrated in Fig. 13), take the DB1 bike at walking distance and then cycle the way to the final destination. In this plan, the weather permits the use of a bicycle; however, if the weather changes during the trip, a new planning request may be triggered by the IMA app.

B. DYNAMIC VIDEO SURVEILLANCE FOR PUBLIC SAFETY

The second use case builds around a *Surveillance App (SAPP)*, which city authorities, e.g., police and fire department, can use for efficient monitoring of an area of interest within the city and coordinating emergency operations. The overall use case flow is depicted in Fig. 15.

As a continuation of the concert event scenario, the authority runs the SAPP app as part of the public safety enforcement around the concert area. SAPP provides situational information resulting from real-time video analytics services, which process video-streams from fixed and mobile camera sensors of both public and private providers, illustrated in Fig. 16. However, SAPP only concerns with the presentation of information supporting the authority's operations, i.e., object detection. It makes use of ISCO city platform as the broker and orchestrator for video-streaming and analytic services. Specifically, SAPP allows authority user to select the area where video-streaming services should be provisioned. The area and required QoS, i.e., availability, access rights, resolution, network capacity, are contained in a service request to ISCO planner component. Based on the requested attributes, the planner queries Service Directory for suitable camera sensors as streaming sources, and available video analytics services. From the query results, the planner creates an optimal composition of the services and provides endpoint information for SAPP to access outputs of video analysis. It should be noted that ISCO does not act as data integration and dissemination platform. It allows point-to-point connectivity of IoT services by providing the services access information of their peers.

The composed streaming and analysis services are also monitored and orchestrated by the ISCO planner, as detailed in Section VI. For example, the quality of a streaming camera may degrade due to changing network conditions. This triggers a replanning process, which identifies a replacement streaming service that can serve the original SAPP request. In our scenario, we assume no nearby camera is available, in which case a drone with camera sensor is selected as the streaming source during replanning. A new goal of the planner, moving the drone to the service area, results in the extension of the previous service chain with route planning service. The calculated route is provided to the drone allowing it to navigate to and capture the footage of the street near the concert area, illustrated in Fig. 17. We note that our implementation currently uses a simulated drone camera service due to practical constraints.

This use case also demonstrates specific security policies being enforced on the ISCO framework by designated city authorities, granting them access to certain public and private services based on situational context.

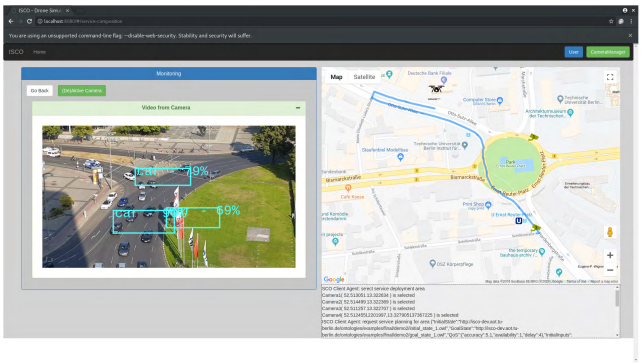

FIGURE 17. Dynamic deployment of a mobile streaming source in Surveillance App.

VIII. SUMMARY AND FUTURE WORK

The ISCO project focused on the initial design and proof-of-concept implementation of a smart city framework that eases the implementation and deployment of dynamic and self-adaptive software. Towards this objective, the general concept and main components of our solution were presented in this article, which consists of a Smart City Object API, an ICN-based distributed service directory, and an adaptive planning component for dynamic service composition. The framework, protocols, and tools developed in our solution can be exploited by various city stakeholders for the efficient creation, discovery, and execution of harmonious services in the smart city service ecosystem. In particular, ISCO eases the functional and qualitative description of city services, as well as their distributed registry and discovery. Two representative use cases, intermodal trip planning and emergency video surveillance, were developed on the prototypical implementation of the ISCO framework. While the current prototype and the use cases have demonstrated the functionality and usability aspects of ISCO, ongoing and future work includes the performance and scalability testing for a larger-scale deployment of all components of the system.

ACKNOWLEDGMENT

A preliminary version of this article was published in the proceedings of CAISE Forum 2018 as a short paper with the title "Service Discovery and Composition in Smart Cities".

REFERENCES

[1] C. Yin, Z. Xiong, H. Chen, J. Wang, D. Cooper, and B. David, "A literature survey on smart cities," *Sci. China Inf. Sci.*, vol. 58, no. 10, pp. 1–18, 2015.
[2] A. Gharaibeh *et al.*, "Smart cities: A survey on data management, security, and enabling technologies," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2456–2501, 4th Quart., 2017.
[3] T. Brandt, B. Donnellan, W. Ketter, and R. T. Watson, "Information systems and smarter cities: Towards an integrative framework and a research agenda for the discipline," in *Proc. AIS Pre-ICIS Workshop-ISCA*, 2016, pp. 1–10.
[4] J. Chen, K. Hu, Q. Wang, Y. Sun, Z. Shi, and S. He, "Narrowband Internet of Things: Implementations and applications," *IEEE Internet Things J.*, vol. 4, no. 6, pp. 2309–2314, Dec. 2017.
[5] M. Chen and V. Leung, "From cloud-based communications to cognition-based communications: A computing perspective," *Comput. Commun.*, vol. 128, pp. 74–79, Sep. 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0140366418302226
[6] M. Chen, J. Yang, L. Hu, M. S. Hossain, and G. Muhammad, "Urban healthcare big data system based on crowdsourced and cloud-based air quality indicators," *IEEE Commun.*, vol. 56, no. 11, pp. 14–20, Nov. 2018. [Online]. Available: https://arxiv.org/abs/1810.10723
[7] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generat. Comput. Syst.*, vol. 29, no. 7, pp. 1645–1660, 2013.
[8] H. Arasteh *et al.*, "IoT-based smart cities: A survey," in *Proc. IEEE 16th Int. Conf. Environ. Elect. Eng. (EEEIC)*, Jun. 2016, pp. 1–6.
[9] *PD 8100 Smart City Overview*, Standard PD 8100:2015, BSI, 2015.
[10] S. G. Araña and M. Menon, "Smart sustainable cities: A guide for city leaders," ITU-T Focus Group on Smart Sustainable Cities, Tech. Rep., 2015.
[11] *Key Performance Indicators Related to the Use of Information And Communication Technology in Smart Sustainable Cities*, document Rec. L.1601, ITU Smart Sustainable Cities, 2015.
[12] *Smart City Concept Model—Guidance for Establishing a Model for Data Interoperability*, Standard ISO/IEC 30182:2017, ISO/IEC, 2017.
[13] C. Manville *et al.*, "Mapping smart cities in the EU," European Parliament, Brussels, Belgium, Tech. Rep. IP/A/ITRE/ST/2013-02 PE 507.480, 2014.
[14] M. A. Razzaque, M. Milojevic-Jevric, A. Palade, and S. Clarke, "Middleware for Internet of Things: A survey," *IEEE Internet Things J.*, vol. 3, no. 1, pp. 70–95, Feb. 2016.
[15] B. Hirsch, T. Konnerth, and A. Heßler, "Merging agents and services—The JIAC agent platform," in *Multi-Agent Programming*. Boston, MA, USA: Springer, 2009, pp. 159–185.
[16] A. L. Opdahl and B. Henderson-Sellers, "Grounding the OML metamodel in ontology," *J. Syst. Softw.*, vol. 57, no. 2, pp. 119–143, 2001.
[17] P. Spyns, R. Meersman, and M. Jarrar, "Data modelling versus ontology engineering," *ACM SIGMOD Rec.*, vol. 31, no. 4, pp. 12–17, 2002.
[18] R. Studer, V. R. Benjamins, and D. Fensel, "Knowledge engineering: Principles and methods," *Data Knowl. Eng.*, vol. 25, nos. 1–2, pp. 161–197, 1998.
[19] N. Guarino, D. Oberle, and S. Staab, *Handbook on Ontologies*. Berlin, Germany: Springer-Verlag, 2009.
[20] N. F. Noy and D. L. McGuinness, "Ontology development 101: A guide to creating your first ontology," Stanford Knowl. Syst. Lab., Stanford, CA, USA, Tech. Rep. KSL-01-05, 2001.
[21] M. Kifer and G. Lausen, "F-logic: A higher-order language for reasoning about objects, inheritance, and scheme," *ACM SIGMOD Rec.*, vol. 18, no. 2, pp. 134–146, 1989.
[22] D. L. McGuinness and F. van Harmelen. (2004). *OWL Web Ontology Language Overview*. [Online]. Available: http://www.w3.org/TR/owl-features/
[23] A. Sutii, T. Verhoeff, and M. van den Brand, "Ontologies in domain specific languages: A systematic literature review," Comput. Sci., Eindhoven Univ. Technol., Amsterdam, The Netherlands, Tech. Rep., vol. 1409, 2014.
[24] M. Sneps-Sneppe and D. Namiot, "On Web-based domain-specific language for Internet of Things," in *Proc. IEEE 7th Int. Congr. Ultra Mod. Telecommun. Control Syst. Workshops (ICUMT)*, Oct. 2015, pp. 287–292.
[25] B. Negash, T. Westerlund, A. M. Rahmani, P. Liljeberg, and H. Tenhunen, "DoS-IL: A domain specific Internet of Things language for resource constrained devices," *Procedia Comput. Sci.*, vol. 109, pp. 416–423, May 2017.
[26] M. Compton *et al.*, "The SSN ontology of the W3C semantic sensor network incubator group," *J. Web Semantics*, vol. 17, pp. 25–32, Dec. 2012.
[27] M. Bermudez-Edo, T. Elsaleh, P. Barnaghi, and K. Taylor, "IoT-Lite: A lightweight semantic model for the Internet of Things," in *Proc. IEEE Int. Conf. Ubiquitous Intell. Comput.*, Jul. 2016, pp. 90–97.
[28] N. Seydoux, K. Drira, N. Hernandez, and T. Monteil, "IoT-O, a core-domain IoT ontology to represent connected devices networks," in *Proc. Eur. Knowl. Acquisition Workshop*. Cham, Switzerland: Springer, 2016, pp. 561–576.
[29] L. Daniele, F. den Hartog, and J. Roes, "Created in close interaction with the industry: The smart appliances reference (SAREF) ontology," in *Proc. Int. Workshop Formal Ontologies Meet Industries*. Cham, Switzerland: Springer, 2015, pp. 100–112.
[30] K. Kotis and A. Katasonov, "An IoT-ontology for the representation of interconnected, clustered and aligned smart entities," VTT Tech. Res. Centre Finland, Espoo, Finland, Tech. Rep., 2012.
[31] D. Pfisterer *et al.*, "SPITFIRE: Toward a semantic Web of things," *IEEE Commun. Mag.*, vol. 49, no. 11, pp. 40–48, Nov. 2011.
[32] S. De, P. Barnaghi, M. Bauer, and S. Meissner, "Service modelling for the Internet of Things," in *Proc. Federated Conf. Comput. Sci. Inf. Syst. (FedCSIS)*, 2011, pp. 949–955.

[33] M. Burstein *et al.* (Nov. 2004). *OWL-S: Semantic Markup for Web Services. Website World Wide Web Consortium*. [Online]. Available: http://www.w3.org/Submission/2004/SUBM-OWL-S-20041122/

[34] J. Domingue, D. Roman, and M. Stollberg, "Web service modeling ontology (WSMO)—An ontology for semantic web services," in *Proc. Position Paper W3C Workshop Frameworks Semantics Web Services*, Innsbruck, Austria, Jun. 2005.

[35] J. Kopecký, T. Vitvar, C. Bournez, and J. Farrell, "SAWSDL: Semantic annotations for WSDL and XML schema," *IEEE Internet Comput.*, vol. 11, no. 6, pp. 60–67, Nov. 2007.

[36] M. Klusch, P. Kapahnke, S. Schulte, F. Lecue, and A. Bernstein, "Semantic Web service search: A brief survey," *Künstliche Intell.*, vol. 30, no. 2, pp. 139–147, 2016.

[37] S. N. A. U. Nambi, C. Sarkar, R. V. Prasad, and A. Rahim, "A unified semantic knowledge base for IoT," in *Proc. IEEE World Forum Internet Things (WF-IoT)*, Mar. 2014, pp. 575–580.

[38] K. Kritikos *et al.*, "A survey on service quality description," *ACM Comput. Surv.*, vol. 46, no. 1, p. 1, 2013.

[39] V. X. Tran and H. Tsuji, "A survey and analysis on semantics in QoS for Web services," in *Proc. IEEE Int. Conf. Adv. Inf. Netw. Appl. (AINA)*, May 2009, pp. 379–385.

[40] E. Giallonardo and E. Zimeo, "More semantics in QoS matching," in *Proc. IEEE Int. Conf. Service-Oriented Comput. Appl. (SOCA)*, Jun. 2007, pp. 163–171.

[41] S. Stevens, "On the theory of scales of measurement," *Science*, vol. 103, no. 2684, pp. 677–680, 1946.

[42] N. R. Chrisman, "Rethinking levels of measurement for cartography," *Cartogr. Geograph. Inf. Syst.*, vol. 25, no. 4, pp. 231–242, 1998.

[43] P. Neirotti, A. De Marco, A. C. Cagliano, G. Mangano, and F. Scorrano, "Current trends in Smart City initiatives: Some stylised facts," *Cities*, vol. 38, pp. 25–36, Jun. 2014.

[44] C. Puleston, B. Parsia, J. Cunningham, and A. Rector, "Integrating object-oriented and ontological representations: A case study in Java and OWL," in *The Semantic Web—ISWC* (Lecture Notes in Computer Science: Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 5318. Berlin, Germany: Springer, 2008, pp. 130–145.

[45] F. S. Parreiras, S. Staab, and A. Winter, "Using ontologies with UML class-based modeling: The TwoUse approach," *Data Knowl. Eng.*, vol. 69, pp. 1194–1207, Nov. 2010.

[46] A. Kalyanpur, D. J. Pastor, S. Battle, and J. A. Padget, "Automatic mapping of OWL ontologies into Java," *Interface*, vol. 4, pp. 98–103, Jun. 2004.

[47] M. Völkel and Y. Sure, "RDFReactor-from ontologies to programmatic data access," in *Proc. 4th Int. Semantic Web Conf.*, 2005, p. 55.

[48] M. Horridge and S. Bechhofer, "The OWL API: A Java API for OWL ontologies," *Semantic Web*, vol. 2, no. 1, pp. 11–21, 2011.

[49] N. Masuch, C. Kuster, and S. Albayrak, "Semantic service manager—Enabling semantic Web technologies in multi-agent systems," in *Proc. Gesellschaft fur Informatik (GI)* (Lecture Notes in Informatics), vol. P-232, 2014, pp. 499–510.

[50] T. Küster, "Eine Methodik zur Entwicklung von Multiagenten-Systemen auf Basis von Geschäftsprozess-Modellen," Ph.D. dissertation, Berlin Inst. Technol., Berlin, Germany, 2017.

[51] S. Cheshire and M. Krochmal, *DNS-Based Service Discovery*, document RFC 6763, Internet Requests for Comments, RFC Editor, Feb. 2013. [Online]. Available: http://www.rfc-editor.org/rfc/rfc6763.txt

[52] S. Cheshire and M. Krochmal, *Multicast DNS*, document RFC 6762, Internet Requests for Comments, RFC Editor, Feb. 2013. [Online]. Available: http://www.rfc-editor.org/rfc/rfc6762.txt

[53] Z. Shelby, K. Hartke, and C. Bormann, *The Constrained Application Protocol (CoAP)*, document RFC 7252, Jun. 2014. [Online]. Available: https://rfc-editor.org/rfc/rfc7252.txt

[54] (2013). *GS1 Object Name Service (ONS) Version 2.0.1, GS1, Ratified Standard*. [Online]. Available: https://www.gs1.org/sites/default/files/docs/epc/ons_2_0_1-standard-20130131.pdf

[55] *CCN Lite: Lightweight Implementation of the Content Centric Networking Protocol*. Accessed: Nov. 30, 2017. [Online]. Available: http://ccn-lite.net

[56] R. Verborgh, T. Steiner, D. Van Deursen, J. De Roo, R. Van de Walle, and J. G. Vallés, "Capturing the functionality of Web services with functional descriptions," *Multimedia Tools Appl.*, vol. 64, no. 2, pp. 365–387, May 2013.

[57] I. Abdullahi, S. Arif, and S. Hassan, "Survey on caching approaches in information centric networking," *J. Netw. Comput. Appl.*, vol. 56, pp. 48–59, Oct. 2015.

[58] Q. Z. Sheng, X. Qiao, A. V. Vasilakos, C. Szabo, S. Bourne, and X. Xu, "Web services composition: A decade's overview," *Inf. Sci.*, vol. 280, pp. 218–238, Oct. 2014.

[59] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, Jan. 2003.

[60] Q. Z. Sheng, X. Qiao, A. V. Vasilakos, C. Szabo, S. Bourne, and X. Xu, "Web services composition: A decade's overview," *Inf. Sci.*, vol. 280, pp. 218–238, Oct. 2014.

[61] Y. Syu, Y.-Y. Fanjiang, J.-Y. Kuo, and S. Ma, "A review of the automatic Web service composition surveys," in *Proc. IEEE Int. Conf. Semantic Comput.*, Jun. 2014, pp. 199–202.

[62] E. Pejman, Y. Rastegari, P. M. Esfahani, and A. Salajegheh, "Web service composition methods: A survey," in *Proc. Int. Multi Conf. Eng. Comput. Scientists* in Lecture Notes in Engineering and Computer Science, vol. 2195, Mar. 2012, pp. 1–5.

[63] Y. Syu, S.-P. Ma, J.-Y. Kuo, and Y.-Y. FanJiang, "A survey on automated service composition methods and related techniques," in *Proc. IEEE 9th Int. Conf. Services Comput.*, Jun. 2012, pp. 290–297..

[64] A. Strunk, "QoS-aware service composition: A survey," in *Proc. 8th IEEE Eur. Conf. Web Services (ECOWS)*. Washington, DC, USA: IEEE Computer Society, Dec. 2010, pp. 67–74.

[65] N. M. Villegas, G. Tamura, H. Müller, L. Duchien, and R. Casallas, "DYNAMICO: A reference model for governing control objectives and context relevance in self-adaptive software systems," in *Software Engineering for Self-Adaptive Systems II* (Lecture Notes in Computer Science), vol. 7475. Berlin, Germany: Springer, 2012, pp. 265–293.

[66] L. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "QoS-aware middleware for Web services composition," *IEEE Trans. Softw. Eng.*, vol. 30, no. 5, pp. 311–327, May 2004.

[67] J. Hoffmann, "FF: The fast-forward planning system," *AI Mag.*, vol. 22, pp. 57–62, Sep. 2001.

[68] J. Pearl, *Heuristics: Intelligent Search Strategies for Computer Problem Solving* (Series in Artificial Intelligence). Reading, MA, USA: Addison-Wesley, 1985.

[69] C. Kuster, N. Masuch, and F. Sivrikaya, "Toward an interactive mobility assistant for multi-modal transport in smart cities," in *Proc. Workshops Service-Oriented Comput. (ICSOC)*, L. Braubach *et al.*, Eds. Málaga, Spain: Springer, 2018, pp. 321–327.

FIKRET SIVRIKAYA received the bachelor's degree in computer engineering from Bogazici University, Istanbul, Turkey, in 2000, and the Ph.D. degree in computer science from the Rensselaer Polytechnic Institute (RPI), NY, USA, in 2007. He was a Research and Teaching Assistant with RPI during his doctoral studies. He served as the Director of the Competence Center Network and Mobility, DAI-Labor, TU Berlin, Berlin, Germany, from 2008 to 2015. Since 2016, he has been the Research Director with the German-Turkish Advanced Research Center for ICT (GT-ARC gemeinnützige GmbH), an affiliated institute of TU Berlin. His research interests include future mobile networks, wireless communications, multi-hop ad-hoc networks, the Internet of Things, and smart cities.

NIZAR BEN-SASSI received the bachelor's and master's degrees in computer science from Technische Universität Berlin, Germany, in 2012 and 2015, respectively, where he is currently pursuing the Ph.D. degree. During his studies, he was a Research Assistant with the Distributed Artificial Intelligence Laboratory, Technische Universität Berlin. Since 2015, he has been a Full-Time Researcher with TU Berlin, with a focus on the research topics of machine learning, AI planning, web service composition, service orchestration, ontologies, and agent technologies.

XUAN-THUY DANG received the Diploma degree in computer science from Technische Universität Berlin, Germany, in 2013, where he is currently pursuing the Ph.D. degree with the Future Mobile Network Group, DAI-Labor. Since 2013, he has been a Researcher with the German-Turkish Advanced Research Centre for ICT (GT-ARC). His main research interests include software-defined networking, cloud computing, service-aware network orchestration, mobile ad hoc networks, delay-tolerant networks, and information-centric networking.

ORHAN CAN GÖRÜR received the bachelor's and master's degrees in electrical and electronics engineering from Middle East Technical University, Ankara, Turkey, in 2011 and 2014, respectively. Since 2014, he has been with DAI-Labor, TU Berlin, and contributing to several research projects, mostly on Robotics and Autonomous Systems. Since 2017, he has also been teaching the Applications of Robotics and Autonomous Systems course, TU Berlin. His research interests include cognitive robotics, artificial intelligence, human–robot collaboration, and multi-agent system modeling.

CHRISTIAN KUSTER received the Diploma degree in computer science from Technische Universität Berlin, Germany, in 2013, where he is currently pursuing the Ph.D. degree. Since 2014, he has been a Researcher with the German-Turkish Advanced Research Center for ICT (GT-ARC), TU Berlin. His research interests include multi-agent systems, agent-oriented software engineering, ontologies, smart cities, and semantic Web.