Service Discovery and Composition in Smart Cities

Nizar Ben-Sassi[1], Xuan-Thuy Dang[1], Johannes Fähndrich[1], Orhan-Can Görür[2], Christian Kuster[1], and Fikret Sivrikaya[1(✉)]

[1] GT-ARC gGmbH, Berlin, Germany
{nizar.ben-sassi,fikret.sivrikaya}@gt-arc.com
[2] DAI-Labor, TU Berlin, Berlin, Germany

Abstract. The ongoing digitalization trend has given rise to the concept of smart cities, targeting the interconnection of city infrastructure and services over a digital layer for innovative technological solutions as well as improvements on existing facilities in cities. This article investigates the critical information system constituents of smart cities that facilitate the holistic integration of its ecosystem and resources with the aim to foster dynamic and adaptive software. We identify three main enablers in this direction: (i) semantic functional description of city objects, representing physical devices or abstract services, (ii) a distributed service directory that embodies available city services for service lookup and discovery, (iii) planning tools for selecting and chaining basic services to compose new complex services. We provide an overview of the approach adopted in our ongoing smart city project for each of these three dimensions.

Keywords: Smart cities · Service discovery · Service composition
Semantic service description · Adaptive planning

1 Introduction

The perennial trend of urbanization has been transforming human life for many decades, whereby the city infrastructure and services become integral parts of our lives in the form of transportation systems, energy systems, and many more. More recently, the rising trend of digitalization brings new dimensions to urbanization; a concept typically captured by the term "smart cities". Advancements in information and communication technologies (ICT), such as the Internet of Things (IoT) and cloud computing provides a foundation for the digitalization of city systems [1]. This often results in better resource utilization among infrastructure providers and more flexible interaction between citizens, authorities and other stakeholders through ubiquitous access to information. However, the abundance of city data makes information management one of the central challenges in enabling cross-domain collaboration. With the lack of unified data and service integration platforms, the transformation to digital cities often results in specific applications that represent isolated service and data silos.

J. Mendling and H. Mouratidis (Eds.): CAiSE Forum 2018, LNBIP 317, pp. 39–48, 2018.
https://doi.org/10.1007/978-3-319-92901-9_4

In general, the information systems of a smart city can be in the form of physical devices with ICT capabilities, entirely virtual online services, or a combination of both, which we commonly refer to as *Smart City Objects (SCO)* in this article. A SCO 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 the transport, environment, and other city domains. The project "Intelligent Framework for Service Discovery and Composition" (ISCO)[1], presented in this paper, aims to develop an open and extensible platform together with supporting tools for the creation and holistic interconnection of SCOs from different sectors and stakeholders. This objective helps move away from fragmented IoT solutions and isolated data silos in cities towards a more integrated and harmonized smart city ecosystem. ISCO enables stakeholders from diverse domains to provide novel, efficient and dynamic services, optionally composed of other existing services in the smart city. One of the main challenges of such a platform is ensuring its scalability while enabling efficient development, deployment, and discovery of smart city objects.

In the remainder of this paper, we present a high-level overview of the ISCO solution (Sect. 2), followed by its main architectural components: a unified semantic model for SCOs based on semantic web technologies (Sect. 3), a scalable distributed architecture for SCO discovery based on information centric networking (Sect. 4), and a two-level planning approach for orchestrating through both generic and application-specific service ontologies (Sect. 5).

2 Background and ISCO Approach

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]. On the other hand, a growing number of smart city initiatives is spawned around the world in recent years [4]. Given the vast breadth and depth of the smart city scope, we do not intend to provide a comprehensive background here, and refer the readers to the surveys on the topic, e.g., [2,5], for a detailed coverage of the socio-technical systems of smart cities. The increasing adoption of digitalization and the existing smart city applications have pointed out several challenges in building such IoT frameworks in cities, ranging from security and network reliability to data management aspects. While there have been many approaches proposed to deal with these challenges, a suitable collaboration scheme between the existing IoT solutions, heterogeneous devices and services remains as an open issue [5].

ISCO Project's approach towards more integrated solutions is to design and develop open platforms and tools for interconnecting heterogeneous entities in a city through what we call *smart city objects (SCOs)*. While the project also covers the networking and security aspects for the interoperability and accessibility of all city objects, our focus in this paper remains on the three core modules of ISCO – represented in Fig. 1 by the SCO API, Service Directory, and

[1] http://www.gt-arc.com/projects/isco/.

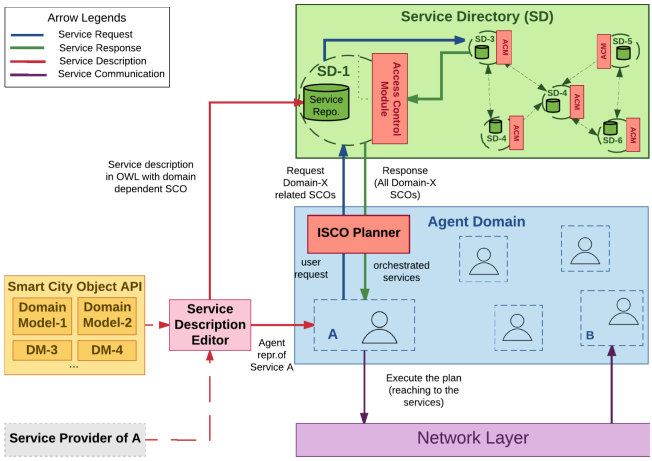

Fig. 1. ISCO architecture: overall interaction of main components

Agent Domain components. In the high-level service workflow of ISCO depicted in Fig. 1, service providers can introduce their service descriptions using our 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. The SD is structured as a dynamic collection of distributed nodes and serves as the repositories and request points for registered services. Every service agent contains an instance of the ISCO Planner to request and orchestrate services. Once a user makes a request, the agent looks up domain-related SCO descriptions distributed in SD infrastructure using an Information-Centric Network (ICN) overlay, as described in Sect. 4. When a set of matching services is returned, the access control module filters out services that are not authorized for the requester. 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. We present the details on the inner-workings and the interactions of these components in the remainder of the paper.

3 Semantic Description of Smart City Objects

For sharing knowledge and modeling SCOs in a huge heterogeneous and dynamic environment, the usage of ontologies is the de-facto approach; domain and data models described in an ontology provide a sophisticated semantic description that can be parsed by an application. Existing approaches that model the IoT domain in a descriptive way commonly adopt the IoT-Lite ontology [6] utilized in FIESTA-IoT[2]. However, this lightweight approach lacks a functional description that can be called by a requester.

[2] http://fiesta-iot.eu/.

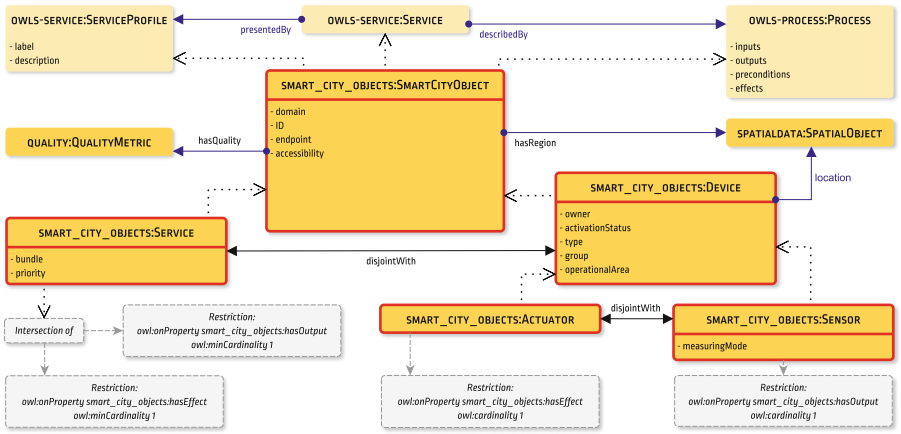

Fig. 2. Centralized overview of the ISCO Smart City Object ontology

On the other hand, for the service-oriented computing community developed well-established standards to describe and invoke functionalities, such as WSDL/SOAP and REST services, but these do not suffice a smart city. Semantic descriptions of software services provide the needed additional information layer, for which several approaches, such as OWL-S [7], WSMO and SAWSDL, have been proposed in recent years, which extend typical service information by preconditions and effects, and define input and output information in a more expressive ontology language, such as OWL. However, these semantic approaches only address web services and not the smart city domain in particular. Our approach extends the OWL-S ontology for smart cities by concepts of IoT-Lite with focus on orchestration. The central class is the abstract *SmartCityObject* class, which is a subclass of the main OWL-S classes; SCO instances are either virtual services or devices, which are further classified into *Actuators* and *Sensors*. The non-functional attributes such as context [8], QoS [9], associated sub domains [10] as well as management attributes enable for sophisticated filtering mechanisms in a smart city with its large amount of functionally overlapping SCOs. An overview of our ontology can be seen in Fig. 2.

4 Scalable Service Lookup and Discovery

Due to 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)[3] and mDNS[4] discovery protocols. Global object clusters are discovered with DNS-SD based on IPv6 and higher level protocols CoAP [11]. In local groups, the multicast mDNS is employed for automatic registration and discovery of devices. The Object Name Service (ONS) [12] 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 IoT-A and BUTLER projects.
- *Service Directory (SD)* Whether domain or geographical discovery are used, 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. 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.

ISCO Service Directory Based on an ICN Overlay. We propose an IoT service directory (SD) in ISCO that self-organizes 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 dependency on discovery mechanism from specific applications and domains.

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. 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

[3] http://www.dns-sd.org/.
[4] http://www.multicastdns.org/.

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. The distributed directory solution takes advantage of the ICN features to design refined 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.

ICN Based Naming Scheme for City Objects. Each SD-Node acts as an ICN router, which serves the requests for SCO's description by its name, or forwards the requests towards other nodes holding the description. Therefore, the design of a naming scheme affects the performance of objects 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 *parameters* part enables the expression of SCO attributes that can be used to look up and discover the SCOs regardless of where they are stored. As shown in Fig. 3, an ICN name of a sensor contains rich semantics describing its domain, location, type, etc. Matching of query attributes and the descriptions is handled by a semantic matcher component in each SD-Node. 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.

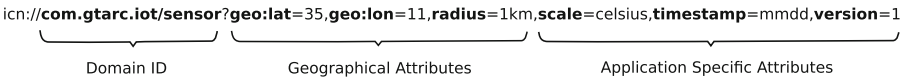

Fig. 3. IoT resource naming scheme in ICN based service directory

Service Directory Node Architecture. The ISCO service directory is constituted by a distributed collection of SD nodes as depicted in Fig. 4. The design of an individual SD-Node, which contains semantic descriptions of SCOs, is shown in Fig. 5. Each functionality is implemented as a modular component: *(1) Triple Store (TDB)* is a component of the Jena[5] project, which serves as a

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

high performance RDF store and query of SCO attributes. *(2) Matcher* component implements mapping methods between requested search attributes and suitable SCO descriptions, which are potential search results. *(3) 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. *(4) 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.

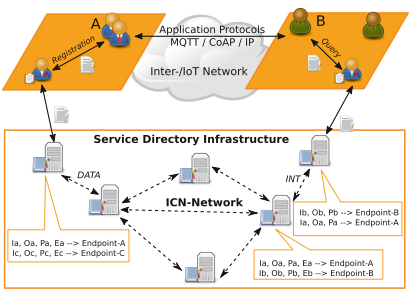

Fig. 4. ICN-based service directory

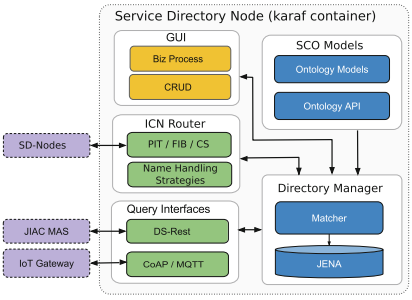

Fig. 5. SD-Node architecture

Scalability. 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 approaches, and strategies for information organization. Figure 4 illustrates a distributed SD infrastructure utilized by the ISCO platform based on a multi-agent system architecture. 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. Moreover, the self-organizing topology allows additional SD-nodes to be added or removed on-demand by applying cloud computing or container technologies (e.g., Docker).

5 Service Composition and Planning

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 [13]

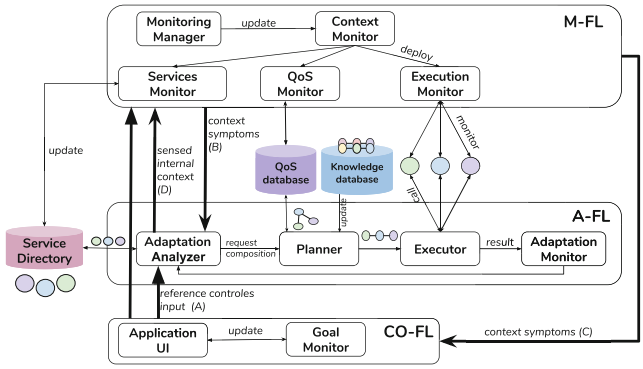

Fig. 6. An overview of the ISCO planning layer components

design guidelines for adaptive systems. This reference model defines three levels of dynamics: the *Objective Feedback Loop (CO-FL)*, the *Adaption Feedback Loop (A-FL)*, and the *Monitoring Feedback Loop (M-FL)*:

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). 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 might be affected by several changes (e.g., goal change, goal is no longer reachable, goal order change, etc.). The system should therefore monitor and evaluate its goals to select the appropriate adaptation.

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 system 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: **(1) Planner** is responsible for combining SCOs by connecting their IOPEs to generate new composite services that satisfy the client application requirements. We extend 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. **(2) 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. **(3) Adaptation Analyzer** 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 (KDB). **(4) Adaptation Monitor** checks the state of the adaption mechanism. The selected adaption needs to change if it is no longer adequate for the current system state. This monitoring is done to observe the performance of the adaption mechanisms.

Monitoring Feedback Loop. Self-adaptive systems need to maintain their context-awareness relevance, in order to adapt at runtime to changing context. The M-FL is the context manager in the DYNAMICO reference model (see Fig. 6). 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. **Services Monitor** 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 correctness 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.

6 Summary and Future Work

We presented the concept of our ISCO framework, a holistic approach for service discovery and composition in smart cities. The current effort focuses on providing an ecosystem that eases the implementation and deployment of dynamic and self-adaptive software. Our ongoing work tackles the common challenges IoT projects face, which are mainly the lack of integrity and interoperability of cross-domain platforms. The unified SCO, the service directory and the service composition and planning layers are the main components of this framework. While our ongoing work focuses on the final stages of development and integration of these components, our planned future work includes the testing of adaptation capabilities and scalability of the system in a heterogeneous dynamic testbed with both physical and simulated entities.

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

References

1. Bibri, S.E., Krogstie, J.: ICT of the new wave of computing for sustainable urban forms: their big data and context-aware augmented typologies and design concepts. Sustain. Cities Soc. **32**(Suppl C), 449–474 (2017)
2. Gharaibeh, A., Salahuddin, M.A., Hussini, S.J., Khreishah, A., Khalil, I., Guizani, M., Al-Fuqaha, A.: Smart cities: a survey on data management, security, and enabling technologies. IEEE Commun. Surv. Tutor. **19**(4), 2456–2501 (2017)
3. Brandt, T., Donnellan, B., Ketter, W., Watson, R.T.: Information systems and smarter cities: towards an integrative framework and a research agenda for the discipline. In: AIS Pre-ICIS Workshop-ISCA 2016 (2016)
4. Manville, C., Cochrane, G., Cave, J., Millard, J., Pederson, J.K., Thaarup, R.K., Liebe, A., Wissner, M., Massink, R., Kotterink, B.: Mapping Smart Cities in the EU. EU Publications Office, Luxemburg (2014)
5. Arasteh, H., Hosseinnezhad, V., Loia, V., Tommasetti, A., Troisi, O., Shafie-khah, M., Siano, P.: IoT-based smart cities: a survey. In: 2016 IEEE 16th International Conference on Environment and Electrical Engineering (EEEIC), pp. 1–6 June 2016
6. Bermudez-Edo, M., Elsaleh, T., Barnaghi, P., Taylor, K.: IoT-Lite: a lightweight semantic model for the internet of things. In: IEEE International Conference on Ubiquitous Intelligence and Computing, pp. 90–97. IEEE (2016)
7. Burstein, M., Hobbs, J., Lassila, O., Mcdermott, D., Mcilraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinivasan, N., Sycara, K.: OWL-S: semantic markup for web services, November 2004
8. Perera, C., Zaslavsky, A., Christen, P., Georgakopoulos, D.: Context aware computing for the internet of things: a survey. IEEE Commun. Surv. Tutor. **16**(1), 414–454 (2014)
9. Kritikos, K., Pernici, B., Plebani, P., Cappiello, C., Comuzzi, M., Benrernou, S., Brandic, I., Kertész, A., Parkin, M., Carro, M.: A survey on service quality description. ACM Comput. Surv. (CSUR) **46**(1), 1 (2013)
10. Neirotti, P., De Marco, A., Cagliano, A.C., Mangano, G., Scorrano, F.: Current trends in smart city initiatives: some stylised facts. Cities **38**, 25–36 (2014)
11. Shelby, Z., Hartke, K., Bormann, C.: The Constrained Application Protocol (CoAP). RFC 7252, June 2014
12. GS1: GS1 Object Name Service (ONS) Version 2.0.1. Ratified Standard 2 (2013)
13. Villegas, N.M., Tamura, G., Müller, H.A., Duchien, L., Casallas, R.: DYNAMICO: a reference model for governing control objectives and context relevance in self-adaptive software systems. In: de Lemos, R., Giese, H., Müller, H.A., Shaw, M. (eds.) Software Engineering for Self-Adaptive Systems II. LNCS, vol. 7475, pp. 265–293. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-35813-5_11