A Collaborative Cyber-Physical Microservices Platform – The SITL-IoT Case

Carlos Gonçalves, A. Luis Osório, Luis Camarinha-Matos, Tiago Dias, José Tavares

A Collaborative Cyber-Physical Microservices Platform – The SITL-IoT Case

Carlos Gonçalves[1], A. Luís Osório[1], Luís Camarinha-Matos[2], Tiago Dias[1], José Tavares[3]

[1]ISEL – Instituto Superior de Engenharia de Lisboa, IPL – Instituto Politécnico de Lisboa, and POLITEC&ID, Portugal, carlos.goncalves@isel.pt, lo@isel.ipl.pt, tiago.dias@isel.pt

[2]School of Science and Technology, NOVA University of Lisbon and CTS-UNINOVA, Portugal, cam@uninova.pt

[3]FORDESI, Informatics Systems, and Solutions Company, jose.tavares@fordesi.pt

Abstract. Managing heterogeneous software and hardware artifacts from multiple suppliers is a complex and challenging process. The integration of sensors, actuators, and their controllers, modeled as IoT elements, also presents significant challenges. Typically, a vendor supplies one or more parts, each one with its proprietary interface, which may raise vendor lock-in and supplier dependencies that can compromise the replacement of some of the artifacts by equivalent ones from competing vendors. The research presented in this paper addresses such challenges in the context of the SITL-IoT project aiming at transforming an industrial agri-food environment towards an open, integrated system-of-systems. We present and discuss a reference implementation of a collaborative platform to simplify the management of different artifacts, supplied by alternative suppliers, modeled as services. More specifically, the concepts of *ISystem (Informatic System)*, *CES (Cooperation Enabled Service)*, and *Service* are used to manage the different elements that compose an agri-food environment transparently and uniformly. We argue that the adopted model simplifies the collaboration among technology suppliers along the life cycle maintenance and evolution of their enabled products.

Keywords: Internet of Things, Systems Integration, Collaborative Networks, Cyber-physical systems, Microservices, Distributed systems.

1 Introduction

Organizations that use different software or hardware elements face challenging problems when updating or upgrading their technological infrastructures. Typically, each technology solution or product is provided by a different supplier with its own proprietary protocols, which quite often makes it very difficult and expensive to replace a given element with an equivalent one from a competing supplier. On the other hand, the Internet of Things (IoT) enables industries to manage their existing sensors and actuators as elements that exist on their local networks or WAN. However, because collaborating suppliers deliver sensors and actuators using different protocols and Application Programming Interfaces (APIs), the integration of such technology artifacts results in complex and demanding processes both in terms of the development and maintenance cycles. Indeed, competing suppliers source elements under technology diversity, raising risks of vendor lock-in or supplier dependencies. Such dependencies compromise the replacement of artifacts, being an obstacle to sustainable innovation.

This paper presents and discusses a reference implementation of an Informatics System of Systems (ISoS) platform [8] that contributes to the Model-Driven Open Systems Engineering (MDEOS) and promotes an open market competitive technology landscape for organizations. The ISoS model establishes a system-of-systems where each system might have market competitors able to provide possible substitutions. The main objective is to make a system, or elements of a system, replaceable by an equivalent technology artifact from an alternative supplier. The notion of Cooperation Enabled Services (*CES*) is adopted as part of the strategy to attain partial substitutability, a challenging endeavor to achieve. The ISoS model comprises three abstraction layers: i) *ISystem,* establishing a coarse computational and cooperation responsibility border; ii) *CES,* as a composite of *Services*; and iii) *Service,* as the operating element that can be a pure software artifact or a cyber-physical element, e.g., an IoT sensor/actuator, as the finer-grained computational responsibility border. By ISoS reference implementation, we mean the instantiation of an operating *ISystem*, named *ISystem*$_0$, aiming to validate and certify the compliance of all the *ISystem*/*CES*/*Service* products.

This work expands further the initial approach of the SITL-IoT project [12], aiming to evolve an industrial agri-food environment towards an agri-food ecosystem supported by an open, integrated system-of-systems. We present the first ISoS reference implementation and discuss its utilization for simplifying the management of artifacts supplied by alternative vendors. Such ISoS reference implementation is the first effort to deliver an actual implementation of the ISoS model, thus allowing organizations to be ISoS enabled. As a case study, we demonstrate the *ISystem*, *CES,* and *Services* instances developed within the SITL-IoT project devoted to structure and manage the agri-food silos environment transparently and uniformly.

The remainder of this paper is organized as follows. Section 2 briefly presents the ISoS background, while Section 3 reviews the SITL-IoT project and its strategies to integrate the ISoS reference implementation. Finally, Section 4 presents the conclusions and discusses future work.

2 Enterprise Architecture with ISoS Background

By adopting the ISoS framework [8], an enterprise platform architecture is based on three core modeling elements: *ISystem*, *CES,* and *Service*. Furthermore, to be ISoS enabled, an organization needs to instantiate the meta-*ISystem*, i.e., an instance of the *ISystem*$_0$, an *ISystem* with the unique role of managing the ISoS landscape. Fig. 1 depicts the primary elements that make an ISoS organization using a SysML Block Definition Diagram. The ISoS abstraction is a composite of exactly one *ISystem*$_0$ and zero or more *ISystems*. Each *ISystem* is composed of one or more *CES*, which are composed of one or more *Services*. The ISoS elements model the technology artifacts through a set of properties, e.g., name, version, supplier, or description. In the case of a *Service*, the modeling element instance has associated the meta-data required for a peer *Service* to access the implemented functionalities.

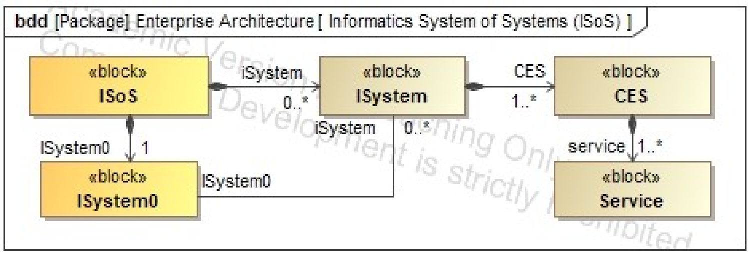

Fig. 1. The simplified SysML block definition diagram of the ISoS model

The ISoS model considers a meta-*element* with management or coordination roles at the ISoS, *ISystem*, and *CES* levels, respectively *ISystem*$_0$, *CES*$_0$, and *Service*$_0$. A primary role of the *ISystem*$_0$ is to act as a directory service managing the metadata of the ISoS elements that exist within an organization. In the current version of the ISoS reference implementation, the *ISystem*$_0$ relies on Apache Zookeeper [4]. Fig. 2 depicts the internal structure of the *ISystem*$_0$ linked to the ISoS Znode, the children nodes *ISystem*$_0$, *ISystem*$_1$, ..., *ISystem*$_N$, the corresponding children *CES*, and, for each *CES*$_J$, the children *Services*. *ISystem*$_0$ has a *CES*$_0$ composed by *Ser*$_0$ and *Ser*$_1$. The ISoS administration user interface has a *CES*$_{UI}$ composed of *Ser*$_0$ and *Ser*$_{UI}$ that makes possible the navigation across ISoS instance elements, facilitating introspection of its properties.

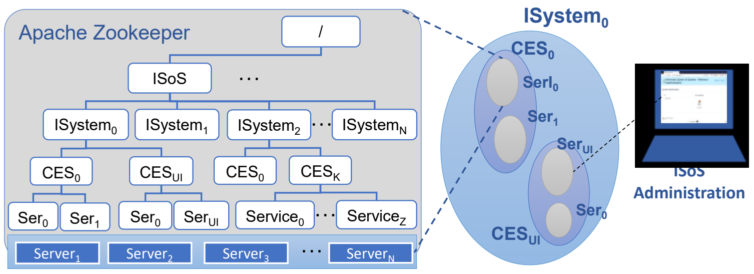

Fig. 2 The internal organization of the *ISystem*$_0$

The adoption of the open-source Zookeeper system is motivated by *ISystem*$_0$ being a critical system since the other *ISystems* depend on its availability. If configured in redundancy mode, the Zookeeper system maintains a consistent replica in N independent servers, preferably based on separate hardware. The approach follows the strategy proposed in [10], considering a reliable *ISystem*$_0$ dependent on the fault-tolerant configuration of the Zookeeper, implementing the Zab distributed coordination algorithm [3], [4]. Furthermore, beyond the fault-tolerance and distributed coordination strategies [6], the *ISystem*$_0$ implementation is prepared to scale several *Service* instances through the Observer nodes concept to speed up read-only service lookup operations [5].

One important feature of the ISoS is its capability to make any *Service* instance accessible both inside and outside the organization. The *ISystem*$_0$ is accessible at *isos.organizationDomain:2058*. Business partners, such as a technology supplier of an *ISystem*, a *CES*, or a *Service*, can use this access to collaborate in the maintenance or evolution of the supplied technology artifacts. The access facility offered by ISoS, through *ISystem*$_0$, is accessible in any business collaboration context by following the appropriate authentication and security mechanisms. In the next section, we detail implementation issues of the reference *ISystem*$_0$ developed in the SITL-IoT project with further contributions from [12].

3 The SITL-IoT Project Case Study

The SITL-IoT research and development project aims at developing an open IoT Bus for cyber-physical elements modeled as *Services*. The project answers the research question of how to evolve towards an open multi-supplier technology landscape. In this section, we show how the ISoS model was used to structure the computing elements that compose the SITL-IoT project.

3.1 The SITL-IoT Base Scenario

Fig. 3 depicts a simplified view of an agri-food company located nearby the seaport of Leixões in the north of Portugal, identified as Organization A.

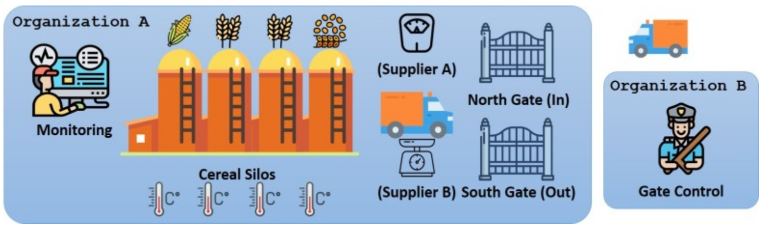

Fig. 3. Case study scenario

In this scenario, we consider only a subset of the elements necessary for loading and unloading cereals to/from trucks for reasons of simplicity. For truck control, access to the industrial facilities is done using two gates: North for inbound and South for outbound. Moving agri-food barges inside the seaport requires an authorization issued by the Port Authority, represented by Organization B. The purpose of the gate in Organization B is to control the trucks discharging the bulk-carrier ships from the seaport area. This area is the Portuguese and EU border with customs and border-police control. As such, the movement of products between the seaport and the agri-food organization requires drivers to authenticate and validate its transport. As shown in Fig. 3, trucks are weighted both inbound and outbound using industrial scales from different suppliers with its own specific weigh controller technology and interfaces. The silos include several temperature sensors that are used to manage the temperature at regularly spaced levels of its structure. This weighing bridge infrastructure, the temperature sensor elements, and other cyber-physical systems of Organization A are modelled as IoT devices. Each IoT device is a *Service* element of the ISoS framework. All *ISystem*, *CES,* and *Service* elements may have an associated synoptic panel for the monitoring and operating of the physical elements. For the visualization of interrelated technology elements, from *ISystems* to *Service*, a generic Synoptics of Things framework is being developed to simplify central supervision interfaces [13].

3.2 The SITL-IoT Project Structure and Elements

The ISoS reference implementation groups the artifacts into specialized projects as Application Programing Interface and Model Elements (APIM), Operations Elements (OPE), Deployment and Operations Elements (DOE), and Monitoring Elements (MOE). This approach aims to facilitate the integratation of complex technology landscapes, complying to the reference structure and following the guidelines suggested by the Collaborative Enterprise Development Environment (CEDE) [7]. Fig. 4 shows the ISoS reference implementation structure with the *ISystem*$_0$ and the corresponding *CES* and *Service* elements. The elements *ISystem* and *CES* are organized using the above-mentioned specialized projects (modules) DOE and MOE, since the APIM and OPE are exclusive of the *Service* elements.

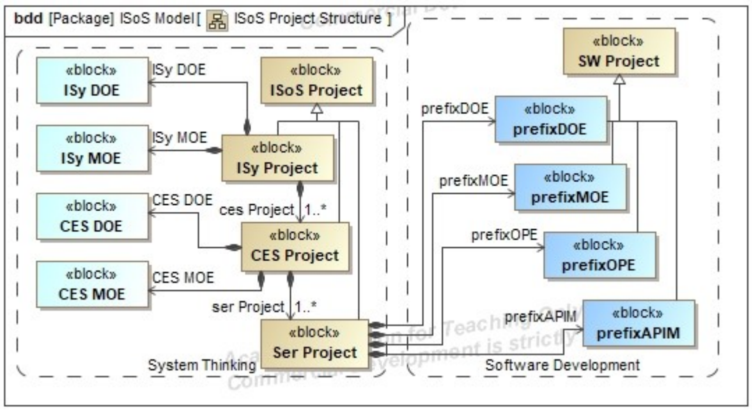

Fig. 4. ISoS Reference Implementation project structure (CEDE concerns)

The DevOps approach inspires the DOE project incorporating the mechanisms to coordinate the development and instantiation of executive parts of ISoS [1]. The MOE aims to deal with the monitoring mechanisms, e.g., by adopting the Simple Network Management Protocol (SNMP) with the respective Management Information Base (MIB) to model the instrumentation of *Service* elements. The technology selection can also use the Java Management Extension (JMX) protocol and the respective instrumentation modeling using Mbean to be managed by JMX agents. The monitoring of technology artifacts is of paramount importance to achieve reliable integrated systems, as discussed in the ISoS framework reliability [11]. OPE organizes the computational logic making a *Service* entity. The project of a *Service* also includes: i) the APIM module, to define interfaces and models specific to the *Service*; ii) the MOE module, to support the implemented monitoring elements; iii) the DOE module, responsible for deploying the *Service*. An alternative is to associate the *CES* DOE module responsible for deploying the composing *Service* elements. Another option is to consider an integrated deployment of an *ISystem* done by its DOE project element. In the current *ISystem* deployment, the strategy is to invoke the DOE projects of the *ISystem* or *CES* composites recursively. For each *CES*, the element executes the deployment logic until the leaf *Service* elements.

The ISoS reference implementation was developed based on the Java ecosystem, using Apache Maven to structure the project, manage the dependencies, and generate isolated and composed artifacts. Nevertheless, very similar principles can be used to develop an ISoS reference implementation using any other technological ecosystem.

As discussed in [7], while ISoS aims to contribute to the substitutability of technology artifacts (*Service*, *CES*, or *ISystem*), technology independence needs to be completed by a unified development environment for unique technology artifacts. Accordingly, the ISoS reference implementation establishes a separation between *ISystems*, *CES*, and *Service* elements as concepts, what we refer to as system thinking to enforce technology independence. The realization of *Services* in some technology and executed within the organization (on-premises) or on the cloud refers to software and deployment/management issues. Fig. 4 depicts *System Thinking* and *Software Development* dotted boxes. The *System Thinking* dotted box represents the ISoS *ISystem*, *CES*, and the *Service* concept as system elements. The *Software Development* box represents the software and integration issues considering the required technology artifacts making the *Service* an executable entity.

As presented and discussed in section 2, the *ISystem*$_0$ primarily acts as the ISoS directory service of an organization (to locate *Service* technology elements). Thus, depending on the current state of *Service* (Deployed, Running, Undeployed, Restarting, Shutting-down, etc.), such state is reflected in the administration interface of the *ISystem*$_0$. The diversity of technologies and strategies to address the DevOps approach, e.g., Ansible, and Kubernetes (container orchestration), motivated a comparative study for a continuous architecting with Microservices and DevOps [14]. Our reference implementation aims to make the *ISystem*$_0$ a governance platform generalized to manage the life cycle of *Service* concept instances and their containment structures (*CES* and *ISystems*). Since a *Service* exists in the context of a *CES* and a *CES* exists in the context of an *ISystem*, we can consider the registering of a *Service* within the ISoS landscape involving the following steps:

a. Create (or update) the meta-information of the corresponding *ISystem*;
b. Create (or update) the meta-information of the corresponding CES(s);
c. Create (or update) the meta-information of the *Service*(s);
d. Start the OPE and MOE modules of the *Service*(s).

As a result, the ISoS reference implementation includes i) a generic *ISystem* DOE capable of implementing step a; ii) a generic *CES* DOE capable of implementing step b; and iii) a generic *Service* DOE capable of implementing steps c and d. Furthermore, since in this case study the Java ecosystem was used as the base for the

ISoS reference implementation, the above artifacts are made available as independent JAR files. This approach allows us to change the implementing artifact by a competing one (substitutability). Although the discussion presented in this work is focused on the Java ecosystem, the proposed concepts can be extended to other ecosystems. In fact, that extension can be a very straightforward process that consists only in the configuration of the above mentioned JAR files to execute native Operating System processes rather than Java processes.

For an *ISystem* reference implementation, the DOE project module is a Java command-line tool (CLI) that receives two XLM files as arguments. The *ISystem* metadata is specified with the argument *-d isystemDef.xml*. The list of configuration elements used to start all the *CES* included in this *ISystem* is set with the argument *-c cesCfg.xml*. Each configuration element has the location of: i) the *CES* DOE module; ii) the file containing the *CES* metadata; and iii) the file containing the configuration of the services included in the *CES*. All file paths in the configuration elements are relative to a base directory, specified as an attribute in the configuration file. Additionally, the configuration file has two attributes to specify the path of the Java Virtual Machine (JVM) and the base working directory of the modules to start.

The *CES* reference implementation considers that the DOE module follows a similar approach to the one used in the *ISystem* DOE module. It is a Java CLI application that receives as arguments the name of the XML file containing the metadata of the *CES* (*-d cesDef.xml*) and the name of the XML file containing the configuration of the *Service* elements that compose the *CES* (*-c serviceCfg.xml*).

The file used to define the *Service* configuration has all the information to start a *Service,* including the DOE, OPE, and MOE modules and the corresponding arguments. Please note that the OPE and MOE modules are the only ones committed to specific functionalities, represented using a darker blue in Fig. 4. The ISoS reference implementation offers a default DOE module, assuming that the OPE and MOE *Service* modules are JAR files receiving their arguments in the command line.

3.3 The ISoS Administration User Interface for the SITL-IoT Case

An administrator can use the ISoS user administration interface to register the different *Services* that compose the ISoS landscape organization using only the OPE and MOE modules of each *Service* and a set of configuration files, as discussed in the previous section. The fulfillment of the ISoS interface with the tree *ISoS*/*ISystem*/*CES*/*Service* is, therefore, a quite straightforward task, as a result of the reference implementation discussed in the previous sections.

It is worth mentioning that advanced abstractions are under evaluation, namely the use of container orchestrations, e.g., the Kubernetes automated container deployment, scaling, and management toolset. However, our approach does not aim exclusively for the cloud. In fact, we strive for a balanced strategy for the organization´s computing technology landscape that can be deployed either on-premises or on the cloud, depending on resource allocation needs and the most advantageous options that can change dynamically. The vendor lock-in risks motivated the proposal of a "… *overlay layer that provides users with an inter-operable and visibility-supported environment* *for MSA-based IoT-Cloud service composition over the existing multiple clouds*" [2]. Nonetheless, the proposed layer seems to introduce additional complexity. The DOE project structuring element can manage the deployment issues in our approach, eventually providing alternative implementations to cope with cloud provider`s heterogeneity.

3.4 Revisiting the SITL-IoT Scenario under a Collaborative Perspective

The ISoS implementation described in the previous sections also enables to analyze the SITL-IoT scenario presented in section 3.1 under the collaborative network perspective. As discussed above, every time a truck needs to enter the agri-food area located on-premises of Organization A, it is necessary to obtain inbound access issued by Organization B. Using the ISoS model and its associated reference implementation, the collaboration between the two organizations is a straightforward process. Each of the gates shown in Fig. 3 is running a *Service*, denoted as *ServiceA*, performing the following actions:

1. Collect the driver and truck identification;
2. Contact the ISoS landscape of Organization B (*isos.organizationB:2058*) to get an instance of its *ISystem*$_0$, denoted as *ISystem*$_0$*B*;
3. Using *ISystem*$_0$*B*, *ServiceA* performs a lookup operation to obtain the *Service* responsible for granting the entry access, denoted as *ServiceGateB*;
4. *ServiceA* uses *ServiceGateB* to authenticate the driver and the truck;
5. If the authentication is successful, the truck can access the agri-food area.

This simple example shows that the presented ISoS reference implementation allows establishing collaboration among two different organizations, each with well-identified responsibilities, without knowing the internal details of the involved organizations. However, the example can be extended to more complex scenarios involving several organizations. The only requirement is that the involved organizations can access the ISoS landscape of each other, i.e., access the involved *ISystem*$_0$. One main problem is that for *ServiceA* of Organization A to access *ServiceGateB* of Organization B, there is a need for *ServiceA* to know a priori the path *ISystem*$_i$/*CES*$_j$/*ServiceGateB* and with it obtain the *ServiceGateB* meta-data. This problem can be resolved using ISoS. With the *ServiceGateB* metadata, the *ServiceA* client from Organization A can get the necessary data to configure the client proxy to access the implemented functionalities properly.

The collaboration infrastructure offered natively by the ISoS framework can be enhanced by adopting the ECoNet collaborative infrastructure [9]. In this case, *ServiceA* of Organization A used its ECoM *ISystem* to have access to a collaboration context shared with Organization B, that provides the required interaction with *ServiceGateB* using an ECoM instance in Organization B. The advantage of collaboration through the ECoM *ISystems* is that domain application *ISystems* share low-level communication, security mechanisms, and higher-level virtual collaboration contexts multi-tenant groups.

4 Conclusions and Further Research

This paper presents and discusses a reference implementation of the ISoS framework, which models the computing technology landscape of an organization. The Java ecosystem adopting the Apache Zookeeper and other open-source projects supports the validation of the framework in the context of the SITL-IoT project. Beyond the *ISystem*$_0$ as a core technological element for any ISoS enabled organization, we present and discuss a project structure to avoid dependency from subcontracted developments. Furthermore, we discuss a modeling schema for the automatic management of ISoS concept instances. Also, we demonstrate how this approach enables configuring operating system services to automatically register an ISoS *Service* and the corresponding *ISystem* and *CES* when the computer (physical or virtual) supporting the *Service*´s execution starts.

We further discuss a monitoring strategy based on SNMP agents operationalized by the ISoS *Service* concept and managed by the MOE project structuring element. The association of monitoring *Service* agents to domain application *Services* requires further research considering the need to abstract legacy protocols, following the adaptive ISoS Service interoperability mechanism.

For software solution providers like Fordesi, ISoS is a tool that brings industrial IoT solutions to the transport and logistics sector. The modularity and decoupling strategies used by the framework enables a quick-wins project management approach that leads to time and cost-effective solutions.

Concerning collaboration issues, the proposed approach based on the ISoS framework offers collaboration support facilities, since services from collaborative organizations can mutually find each other and interoperate based on the I_0 canonical entry point and ISoS metadata facilities. In addition, we discuss the alternative ECoNet using the collaborative contexts and virtual collaboration contexts as shared infrastructure elements. While the collaboration mechanisms offered by ISoS proved to be sufficient for the current business case, further research will validate the adoption of ECoNet infrastructure as a more general approach.

Acknowledgments. The research conducted by GIATSI/ISEL/IPL was developed in collaboration with the SOCOLNET scientific network and its ARCON-ACM initiative. The ANSR/SINCRO, BP/HORUS, and FORDESI/SITL-IoT-PT-2020 projects partially supported this research. Partial support also from the Center of Technology and Systems – UNINOVA, and the Portuguese FCT Foundation (project UIDB/00066/2020), and the European Commission (project DiGiFoF). We also recognize the excellent contributions from Bruno Serras as a research fellow.

References

1. A. Balalaie, A. Heydarnoori, and P. Jamshidi. Microservices Architecture Enables DevOps: Migration to a Cloud-Native Architecture. *IEEE Software*, 33(3):42–52, May 2016.

2. J. Han, S. Park, and J. Kim. Dynamic OverCloud: Realizing Microservices-Based IoT-Cloud Service Composition over Multiple Clouds. 9, 2020.
3. P. Hunt, M. Konar, F. P. Junqueira, and B. Reed. ZooKeeper: Wait-Free Coordination for Internet-Scale Systems. In *Proceedings of the 2010 USENIX Conference on USENIX Annual Technical Conference*. USENIX Association, 2010.
4. F. P. Junqueira and B. Reed. *ZooKeeper: Distributed Process Coordination*. O'Reilly Media, Inc., 1st edition, 2013.
5. F. P. Junqueira, B. Reed, and M. Serafini. Zab: High-Performance Broadcast for Primary-Backup Systems. In *Proceedings of the 2011 IEEE/IFIP 41st International Conference on Dependable Systems & Networks*, pages 245–256. IEEE Computer Society, 2011.
6. L. Lamport. Paxos Made Simple. *ACM SIGACT News (Distributed Computing Column) 32*, pages 51–58, December 2001.
7. A. Luís Osório. Towards Vendor-Agnostic IT-System of IT-Systems with the CEDE Platform. In: *Collaboration in a Hyperconnected World*. PRO-VE 2016. IFIP AICT, vol 480. Springer, Cham. https://doi.org/10.1007/978-3-319-45390-3_42
8. A. Luís Osório, Adam Belloum, Hamideh Afsarmanesh, and Luís M. Camarinha-Matos. Agnostic Informatics System of Systems: The Open ISoS Services Framework. In: *Collaboration in a Data-Rich World*, pages 407–420. PRO-VE 2017. IFIP AICT, vol 506. Springer, Cham. https://doi.org/10.1007/978-3-319-65151-4_37
9. A. Luís Osório, Luís M. Camarinha-Matos, and Hamideh Afsarmanesh. ECoNet Platform for Collaborative Logistics and Transport. In: *Risks and Resilience of Collaborative Networks*. PRO-VE 2015. IFIP AICT, vol 463. Springer, Cham. https://doi.org/10.1007/978-3-319-24141-8_24
10. A. Luís Osório, Luís M. Camarinha-Matos, Hamideh Afsarmanesh, and Adam Belloum. On Reliable Collaborative Multimodal Services. In: *Collaborative Networks of Cognitive Systems*. PRO-VE 2018. IFIP AICT, vol 534. Springer, Cham. https://doi.org/10.1007/978-3-319-99127-6_26
11. A. Luís Osório, Luís M. Camarinha-Matos, Hamideh Afsarmanesh, and Adam Belloum. Liability in Collaborative Maintenance of Critical System of Systems. In: *Boosting Collaborative Networks 4.0 . PRO-VE 2020, Valencia, Spain, November 23-25, 2020, Proceedings*, volume 598 of *IFIP AICT*, pages 191–202. Springer International Publishing, 2020. https://doi.org/10.1007/978-3-030-62412-5_16
12. A. Luís Osório, Luís M. Camarinha-Matos, Tiago Dias, and José Tavares. Adaptive Integration of IoT with Informatics Systems for Collaborative Industry: the SITL-IoT Case. In: *Collaborative Networks and Digital Transformation*. PRO-VE 2019. IFIP AICT, vol 568. Springer, Cham. https://doi.org/10.1007/978-3-030-28464-0_5
13. Bruno Serras, Carlos Gonçalves, Tiago Dias, and A. Luís Osório. Synoptics of Things (SoT): An Open Framework for the Supervision of IoT Devices. In *To appear in 5th International Young Engineers Forum on Electrical and Computer Engineering*. IEEE Xplore digital library, 2021.
14. D. Taibi, V. Lenarduzzi, and C. Pahl. Continuous Architecting with Microservices and DevOps: A Systematic Mapping Study. *CoRR*, abs/1908.10337, 2019.