

Received October 22, 2021, accepted November 9, 2021, date of publication November 10, 2021, date of current version November 17, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3127472

SuDaMa: Sustainable Open Government Data Management Framework for Long-Term Publishing and Consumption

ELENA SÁNCHEZ-NIELSEN¹, ALEJANDRO MORALES^{1,2}, OMAR MENDO^{1,2}, AND FRANCISCO CHÁVEZ-GUTIÉRREZ^{1,2}

¹Departamento de Ingeniería Informática y de Sistemas, Universidad de La Laguna, 38200 San Cristóbal de La Laguna, Spain

²Unidad TIC, Parlamento de Canarias, 38002 Santa Cruz de Tenerife, Spain

Corresponding author: Elena Sánchez-Nielsen (enielsen@ull.edu.es)

This work was supported in part by the Spanish Ministry of Science and Innovation under I+D+i Contract PID2019-107228RB-I00, in part by the Ministry of Economy, Knowledge, and Employment of Canary Islands, and in part by the European Social Funds (ESF) through Canary Islands 2014–2020 Strategy Aim 3 under Project ProID2021010012.

ABSTRACT Sustainable open data management systems are key elements to overcome long-term publishing and consumption challenges on open data platforms. However, most existing solutions have not been envisaged as sustainable approaches to cope with key data attributes such as timeliness, accessibility and usability. In this paper, we present a framework that addresses these challenges and ensures the efficient publication of dynamic data on open data platforms, as well as improving the consumption experience of end-users. The framework describes the architecture of a sustainable system that can operate continuously in an automated way. It involves: 1) introducing an evolvable and scalable ecosystem to guarantee the access to dynamic data when they are made available; 2) tackling governance using an autonomous agent to provide the capacity of dynamically publishing/unpublishing data resources on open data platforms. It ensures easier accessibility and greater efficiency for data developers, facilitating the development of data-enabled products, thus helping developers to seize the opportunities of the data economy; and 3) improving the consumption experience by introducing conversational bots, which enhances the usability of open data. We demonstrate the results and validity of the solution in practice through its implementation, evaluation and exploitation as a practical system in the Parliament of the Canary Islands, Spain. The framework developed can be considered a generic solution to manage open data publishing and consumption challenges in other domains.

INDEX TERMS Open data framework, open data sustainability, dynamic open data, open data publishing, open data consumption.

I. INTRODUCTION

Open government data (OGD) refers to government-related data that is made available in open and reusable machine processable formats to the public and can be easily accessed, freely used, and shared by anyone for any purpose [1], [2]. It includes different data such as parliamentary data, budget and spending, census, geographical data, as well as data related to climate, public transportation, traffic and education.

Nowadays, OGD is among the most published open data on the Web and is continually increasing in terms of volume over time. This enables data consumers to use these

data for better policymaking, greater innovation, economic growth, and societal progress in general. To achieve these potential benefits of OGD, it is essential to overcome different challenges that encompass a wide range of issues from social to technical [3].

From the technological perspective, according to the OGD lifecycle [2], OGD publishing and consumption are identified as the challenging processes that must be addressed to unlock the potential benefits of OGD. Moreover, both processes should be envisaged from a long-term viewpoint to obtain the full potential of OGD [4], [5]. Different assessment frameworks in the literature [6], [7]–[13] as well as European regulations [14]–[16] have identified timeliness, accessibility and usability as key data attributes that should

The associate editor coordinating the review of this manuscript and approving it for publication was Liang-Bi Chen.

be addressed to ensure long-term open data publishing and optimal consumption when OGD solutions are being designed and developed from a technological focus.

To date, the degree of maturity of technological solutions has evolved in line with the potential benefits of OGD, the needs of open government initiatives, and the increasing volumes of available data. Following this path, the technological focus is shifting to ensure long-term OGD publishing and consumption. Referring to the need for OGD to be easily accessible, usable and released continuously.

Ensuring the continuous updates of data is particularly important for the publication of dynamic data (i.e., data that is updated as new information becomes available), since the economic and political value of OGD depends on the immediate availability of new information and regular updates. On the other hand, enhancing the OGD usability (i.e., how easily can the published data be used) is also essential to guarantee the full potential of data consumption by data consumers.

Currently, most technological existing solutions have not been envisaged as sustainable approaches (i.e., systems that are able to last in an automated way) to sustain long-term OGD publishing and consumption. The main reasons are that opening up data around the world, in many cases, originated as part of a politically driven open data initiative. Also, it is in response to the emerging Open Government Partnership (OGP) movement that needed to be implemented within a set timeframe [6], using existing open data platforms (ODPs) [3] as key components to enable data publishers to publish and share their datasets as open data, without considering the key data attributes to ensure long-term publishing and consumption (i.e., timely, accessible and usable data). Following the proliferation of these ODPs as data infrastructures for OGD publishing and consumption, the technological solutions have focused on two approaches. On the one hand, the development of custom ODPs for specific purposes and, on the other hand, the development of underlying data management systems as well as their integration with existing ODPs. However, to date, both approaches have not been envisaged as viable and sustainable solutions (i.e., solutions that are sustainable in an automated way) to ensure long-term publishing and overcome consumption challenges (i.e., continuous publication of dynamic data, easy access for developers as well as improving the consumption experience of end-users).

Providing data management system solutions in the domain of eGovernment also requires addressing properly current regulations and digital government policy documents. In the European Union (EU) context, it includes the new open data directive [14], the communication related to the new EU strategy for data [15], and the communication associated with the common EU data space [16]. They all focus on:

- Adopting application programming interfaces (APIs) as mechanisms to facilitate the access to data opened for reuse.

- Using APIs based on several principles: availability, stability, maintenance over lifecycle, uniformity of use and standards, user-friendliness, and security.
- Providing real-time access to dynamic data, meaning that the public sector bodies should make data available for reuse immediately after collection by ways of suitable APIs.

Thus, the aim of this work is to present SuDaMa, which is an acronym for *Sustainable Open Government Data Management Framework* for long-term OGD publishing and consumption. It addresses the above-mentioned challenges to provide a sustainable data management system.

When reading this work, it is important to keep in mind that when we refer to data consumers, we are referring to both end-users and developers. Moreover, when we refer to end-users, we are referring to individuals that access OGD. This is distinct from developers and organizations that develop data-enabled products from OGD.

Bearing all the above in mind, when constructing SuDaMa, we prioritized how to automate the publication of OGD from data systems that operate continuously and, how OGD could be accessed in an easy and more usable way by data consumers. Consequently, the four main contributions of our work are to:

- Present an OGD management framework as a holistic solution for long-term publishing and consumption. This framework comprises the different phases from set-up to post-deployment. That is, design of the system's architecture, development and deployment of the system, as well as its evaluation and exploitation in a real-world OGD scenario. To do this, we have collaborated with the Canary Islands Parliament,¹ in all the mentioned phases.
- Introduce an evolvable and scalable API ecosystem, as main component of the system's architecture, to ensure real-time access to dynamic data as well as to generate dynamically new datasets when they are available. This not only eliminates the tedious and manual process of developing and maintaining API resources for data publishers, but also enables sustainable development for developers, providing data consistency and ensuring greater efficiency and scalability for their data-enabled products. This contribution involves: 1) formalizing all the conceptual elements of the ecosystem, 2) addressing the design of API resources and the development of techniques that ensure the dynamic generation of API resources for variable data collections, which are not known ahead of time and evolve over time, and 3) automating the governance of the ecosystem by an autonomous software agent to provide the capacity of dynamically publishing/unpublishing data resources on ODPs.
- Offer advanced ways of consuming data between end-users and data publishers by introducing an OGD

¹<https://www.parcn.es>

API-driven bot as a conversational interface. This provides user-friendly access to end-users when data are large and diverse, improving the usability of OGD. This contribution involves developing techniques of how best to handle the interaction with end-users when addressing the information needs from OGD and, translating them into effective queries for the resulting API ecosystem to build up an answer.

- Implement the proposed solution and evaluate the two core components of the system's architecture (i.e., the API ecosystem and the OGD API-driven bot) by a Verification, Validation and Testing model in a real-world OGD scenario. This contribution shows the results obtained in practice and the value created for researchers and practitioners in real-life problems.

The remainder of the paper is structured as follows. Section II summarizes the background. Section III outlines the methodology applied. Section IV describes the requirements to design the architecture of the sustainable data management system to ensure long-term OGD publishing and consumption. Section V presents the proposed architecture and describes the novel aspects of the core components to achieve the requirements. Section VI evaluates the main aspects of the core components of the architecture according to the specified requirements with a practical system for the Canary Islands Parliament. Finally, Section VII highlights the conclusions obtained from real-world settings.

II. BACKGROUND

A. LONG-TERM OGD PUBLISHING AND CONSUMPTION

OGD is expected to bring not only economic and social benefits, but also advanced forms of policy decision-making through data analytics and knowledge sharing capabilities for various stakeholders in both the public and private sectors [18]. According to the OGD lifecycle, OGD publishing and consumption are identified as the challenging processes that must be addressed to take full advantage of the benefits of OGD [2].

OGD publishing refers to the specific process of making data widely accessible by publishing it on government portals. Therefore, maintaining OGD over time is vital to ensure the long-term sustainability of published OGD.

OGD consumption refers to the process of use and reuse of such data. Data consumption can be either data exploration, where an end-user visualizes or scrutinizes OGD, or data exploitation, where developers add value to OGD by carrying out the development of data-enabled products.

Both processes must be envisaged from a long-term viewpoint to take full advantage of the benefits that OGD offer (i.e., promoting transparency, accountability and participation by governments as well as fostering greater innovation, economic growth and societal progress in general) [2].

Different assessment frameworks in the literature have identified that to cope with long-term data publishing and consumption challenges, three key data attributes must be addressed: timeliness, easy data access and

usability [6], [10]–[16]. This is attributed to the fact that inconsistent or outdated data as well as inefficient data accessibility and usability leads to poor data quality, which potentially hampers their efficient reuse to generate political, economic and social value [2], [13], [19]. Each of these three critical factors is discussed below.

1) TIMELY DATA

Timely data refers to whether data are made available to the public immediately after the data are created as well as when the data are updated. It plays an essential role in the publication of dynamic OGD, preventing subsequent issues related to data consumption and guaranteeing an efficient reuse of the data.

2) ACCESSIBLE DATA

Accessible data refers to how data is made available and can be retrieved. Making OGD accessible to developers is a crucial point that needs to be addressed by data publishers to facilitate the development of data-enabled products. It introduces the challenge of how developers can easily access available datasets. The most commonly used retrieval method has been bulk data: meaning that the complete dataset should be available in downloadable form from data catalogues.

More recently, APIs as access methods to datasets have been recognized as being fundamental for an emerging data economy [20]. This has also been highlighted from the EU by different European digital government policy documents and initiatives. It includes the EU *Open Data Directive* [14], which specifically requires the mandatory use of APIs for high-value and dynamic datasets as conditions for their reuse. The *European Study for Data* [15] reports on the future investment into EU-wide interoperable data spaces; and the communication *Towards a Common European Data Space* [16] points out the use of APIs for simpler and more automated access to and use of datasets.

APIs have long been a cornerstone of information and communication technology architectures. They are machine-to-machine digital interfaces that facilitate the exchange of data and services [20]. REpresentational State Transfer (REST) has become the prominent architectural style for designing Web APIs due to its lightweight nature, adaptability to the Web, and scaling capacity [21]. A Web API using the REST architectural style is referred to as a REST API. They are designed around resources, which are any type of object, data or service.

REST architectural style consists of a set of constraints to address such factors as visibility, reliability, scalability and performance. In general, REST APIs describe four interface constraints, namely: 1) Identification of resources (i.e., use Uniform Resource Identifier (URI) to identify resources); 2) Manipulation of resources through representations (i.e., representations are transferred between REST components to manipulate resources, e.g., use JSON as exchange format); 3) self-descriptive messages (i.e., enable intermediate

processing by constraining messages to be self-descriptive); and 4) hypermedia as the engine of application state (i.e., resources link to each other in their representations using hypermedia links and forms).

The increasing adoption of APIs has triggered the creation of standards to formally describe a REST API in terms of its resources and operations. The most representative standards are: 1) Open API Initiative (OAI), formerly known as Swagger [22] that is based on a JSON data representation and, 2) RESTful API Modeling Language, RAML [23] which is based on human-readable data serialization language.

3) USABLE DATA

The usability of data has become a significant feature to guarantee OGD consumption for end-users. It addresses how easily the published data can be used [2].

Despite the different OGD initiatives, access to OGD by end-users is not as high as expected [24]–[27]. This can be partially attributed to the fact that often a non-user-friendly interface has been built on the top of ODPs for access and navigation. Basically, visual representations or large tabular files are constructed from the data and provided as consumption sources. This data consumption approach, in many cases, is not easy to consult, particularly when data are large and diverse. This results in a decrease in the usability of the data.

B. IMPLEMENTATION OF OGD PUBLISHING

The complexity of software implementation has evolved in line with the potential benefits of OGD, the needs of open government initiatives, and increasing data volumes. In this context, the software implementation adopted by data publishers to manage OGD publishing can be classified into three approaches: 1) Using existing ODPs, 2) Development of Custom ODPs, and 3) Development of Data Management Systems. In the following subsections, each one of these approaches are described and discussed.

1) USING EXISTING ODPS

Many governments have made their OGD available through data portals, such as the U.S. Government's Open Data,² Australian Open Government Data,³ Canada Open Government Portal,⁴ Swiss Open Government Data,⁵ Berlin Open Data,⁶ Spanish Open Government Portal⁷ and Danish Open Data.⁸ Many of these portals have been built on top of an open-source data platform, such as CKAN,⁹ which provides a data infrastructure to publish and share data with basic search capabilities, visualization options and associated metadata to

assist access to these data. Other known ODPs are DKAN, Junar, Socrata, Enigma, PublishMyData, OpeDataSoft and Zenodo [3]. All these platforms are key elements used by data publishers to publish and share their OGD as well as mediate public access to data. Although these platforms provide a number of data management services, data browsing and content management services, these platforms are not viable solutions to address the up-stream management of data production to ensure long-term OGD publishing and consumption in terms of timely data, accessible data and usable data. This means that data publishers are responsible for developing their own underlying management data systems as well as their integration with existing ODPs. However, many of the existing solutions have not addressed the development of these aspects, as has been revealed in detailed studies conducted by [1], [2]. Results obtained from these studies show that only about the half of the datasets in the portals analyzed were updated according to their schedule and the nature of the contained data. The main reasons are that opening up data around the world, in many cases, originated as part of a politically driven OGD initiative or in response to the emerging OGP movement that needed to be implemented within a set timeframe [6]. This situation has potentially hampered an efficient reuse of OGD, limiting therefore the promised benefits of OGD to the society and economy [2].

2) DEVELOPMENT OF CUSTOM ODPS

Over recent years, several ODPs have been designed and developed with distinct features for specific purposes and presented to the research community. Among them are a linked data platform, *QuerioCity*, to publish, search and link city data from static datasets or stream data from sensors [28], *AECIS* to automate data discovery and urban stream data integration [29], a streamlined process to collect, store and disseminate monitored data in an urban environment [30], a *City Data Pipeline* to seamlessly access the data from other data providers [31], *VIVO* to populate research data [32], and *LinkedLab* as a platform to manage data from research communities [33]. All these ODPs are capable of dealing with OGD with different affordances, however they were not envisaged to cope with the key data attributes (i.e., timely data, accessible data and usable data) to ensure long-term OGD publishing and consumption.

3) DEVELOPMENT OF DATA MANAGEMENT SYSTEMS

Although there are numerous OGD initiatives that use ODPs such as CKAN to publish their OGD, there still exist a number of drawbacks which prevent them from reaching the full potential of OGD. One of the main reasons is that publishing OGD not only involves the act of making the data available on data portals, but also maintaining them over time to guarantee their efficient use and reuse. To address this issue, the development of underlying data management systems and their integration with existing ODPs has been proposed. To date, however, there are some factors that have hampered publishers attempting to carry out this approach. The main

² <https://www.data.gov/>

³ <https://www.australia.gov.au/>

⁴ <https://open.canada.ca/en>

⁵ <https://opendata.swiss/en/>

⁶ <https://daten.berlin.de/>

⁷ <https://datos.gob.es/>

⁸ <https://www.opendata.dk/city-of-copenhagen>

⁹ <https://ckan.org/>

reason is that most OGD portals were created quickly as part of a politically driven OGD initiative to be implemented in a limited timeframe. Thus, most software solutions were set up without considering the data update maintenance required over time by data publishers.

However, given the importance of achieving the full benefits of OGD, new software solutions considering the development of data management systems are emerging. For instance, in the context of budget and spending data, a data management system has been proposed to address publishing and exploration [34], whilst in the smart city context, a data management system to streamline data management and publishing has been described in [35]. Although these proposals present an approach to deal with timely data, both tackle the challenge of managing, sharing and publishing data by using Extract-Transform-Load (ETL) technology [36]. Although popular, this method is subject to several weaknesses which prevent it from achieving long-term OGD publishing. Among the main drawbacks are: 1) the overall processing time of data gathering is typically not near real-time, 2) the manual development effort to make every data release available can be very time-consuming, which increases software development and maintenance costs, thus incurring considerable expenses for the organizations, and 3) data updates are available according to a scheduling strategy, limiting their availability as soon data is available.

In summary, most approaches based on portal data and data management systems have been quickly created without considering the data maintenance required over time as well as not considering the sustainability of solutions. That is, the capability of data management systems and their ability to evolve dynamically in an automated way to ensure long-term OGD publishing.

C. IMPLEMENTATION FOR OGD CONSUMPTION

A very relevant challenge to achieving the full potential of published OGD on ODPs is their use by data consumers. Although OGD is associated with multiple benefits, several studies found many challenges to using OGD [18]. One of the most widely discussed challenges is that end-users have a hard time making sense of raw data since visual representations or large tabular files are constructed from the data and provided as consumption sources. The researchers suggest that only a small number of end-users are capable of understanding the underlying statistical meanings and implications of OGD and fully realize its benefits. To address this challenging issue, the formalization of a platform, namely *MODA*, was designed to provide a systematic value-creation process that helps stakeholders identify the most suitable information assets and convert them into forms that can be easily consumed [37]. However, only a conceptual approach has been introduced, neither its implementation nor findings have been addressed yet. Other solutions have been proposed for exploring datasets by web-based user interfaces in a more user-friendly way, such as in the educational

context [38], budget and spending context [34], and in health program initiatives [18].

However, more recently, the development of new digital channels of interaction based on natural language processing have been proposed as a key enabler to foster access to and querying of open data. In this context, conversational bots are useful Artificial Intelligence tools that can be exploited as a more natural way to enable end-users to find and compose the information needed in different domains. Formally, bots are intelligent agents, defined as devices that perceive their environment and take actions that maximize their chance of success at achieving some goal [39]. They can understand a spoken language and use speech communication as the user interface [40], [41].

The adoption of bots in the public sector as a new way of interaction poses two issues. On the one hand, the development of a knowledge base from government experts, and, on the other, its integration with dialog management.

Although the adoption of conversational bots has recently been proposed as a way of transforming the communication between citizens and government [42], little attention has been given to developing it for open data to create a user interaction that is as natural as possible for querying and retrieving data.

The conclusion of the review of the state of the art tends to suggest that to exploit the benefits of OGD, from the technological viewpoint, it is essential to invest in making good design decisions in the development of new data management systems based on the key data attributes (i.e., timely data, accessible data, and usable data) as well as making sustainability explicit.

Thus, the aim of this paper is to present a novel sustainable data management system. That is, a data management system that can last over time with the ability to evolve dynamically in an automated way to ensure long-term publishing and consumption of dynamic OGD. The main contributions of the paper can be summarized as: 1) the formalization of an evolvable and scalable API-enabled ecosystem to provide the automation of timely data and easy data accessibility, 2) the autonomous governance of the API ecosystem to ensure the sustainability of the system, 3) to offer advanced ways of consuming data to improve data consumption, and 4) to validate the propose solution in a real OGD scenario to show the findings obtained in practice with real-life problems.

III. METHODOLOGY

The proposed framework, SuDaMa, has been created in the context of an open data innovation project between researchers of University of La Laguna (ULL) and practitioners from the Parliament of the Canary Islands with the methodology shown in Figure 1. This methodology follows the steps of scientific research in the design of information systems [43]: 1) problem identification and motivation, 2) define the objectives for the solution, 3) design and development, 4) evaluation, and 5) demonstration.

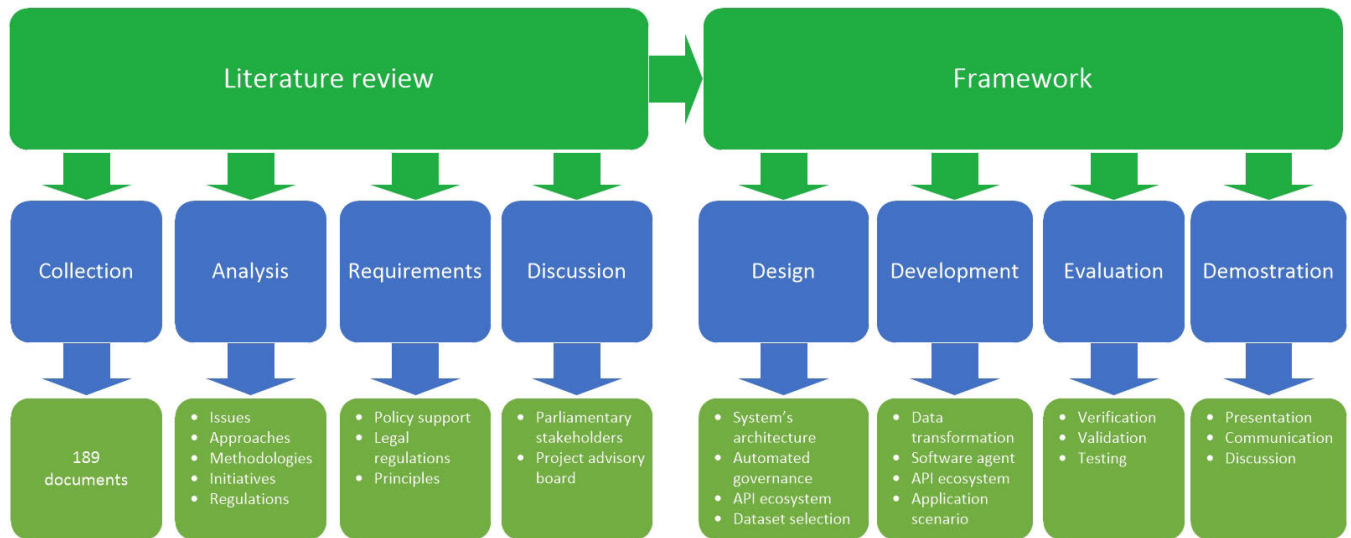


FIGURE 1. Overall methodology for the data management framework.

Given our goal is also to guarantee the sustainability of the solution, we added an additional step named exploitation. This allowed us to verify the evolution of the results over time after the solution was deployed. Hence, the methodology (see Figure 1) began with a literature review involving academic papers, official documents and best-practice standards that covered the different insights and challenges facing open data publishing and consumption and considering technical, legal, political, operational and economic angles. Next, from the literature review, the main issues were determined that are not covered by current data management systems to ensure OGD sustainability in real OGD settings. Then, the requirements for problem-solving were identified considering existing policy support, legislation and technical principles. From this, a framework was designed and discussed with parliamentary stakeholders and the project advisory board. The resulting framework consisted of four stages. First, the design encompassed the formalization of the system's architecture and dataset selection. Second, the development of the system for each of the different components was accomplished. Third, the system was evaluated by a verification, validation and testing-based model. Fourth, a demonstration of the result was presented as an open data initiative of the Parliament of Canary Islands on parliamentary transparency. It included presentation, communication and discussion with media and participants. An OGD driven application was also implemented as way to validate the OGD consumption experience. Nowadays, the deployed solution is being used and continually validated by the Parliament.

IV. LONG-TERM OGD PUBLISHING AND CONSUMPTION REQUIREMENTS

In this section, we determine the requirements to design and develop the architecture of a sustainable data management system to ensure long-term OGD publishing and consumption.

According to the key aspects discussed in Section II-A, by long-term OGD publishing and consumption we mean to cope with three key data attributes: timely data, accessible data and usable data. The two first requirements are essential to ensure long-term publishing, whilst the third requirement is necessary to enable long-term consumption.

As a result, design decisions on the architecture must focus on how to develop its components considering these three requirements, so that the system works in an automated way. For its application in real-world OGD scenarios, the solution proposed must also be fully aligned with the new open data directive [14].

A. REQUIREMENT 1: TIMELY OGD

This requirement is focused on providing dynamic OGD publishing in a timely way. The system must be able to address both dynamic data and datasets from organizations that evolve over time according to their main functions. By dynamic data, we mean that data is made available to the public at real-time after the data is created or up to date. By dynamic datasets, we mean the creation of new datasets that are not known a priori and evolve over time.

B. REQUIREMENT 2: ACCESIBLE OGD

To ensure sustainable and easy data access to developers, the system must be designed as an enabler to unify access to all data, allowing developers to access and integrate them into their applications. As a result, the system must deliver a continuous flow of value to developers, ensuring that the outputs are updated and synchronized with any changes in the datasets.

The resulting system must also be connected to ODPs and not only accessible through data publishers' infrastructures to foster greater data accessibility, as well as complying with the principles addressed by EU legislation by adopting APIs as mechanisms to improve access to open data.

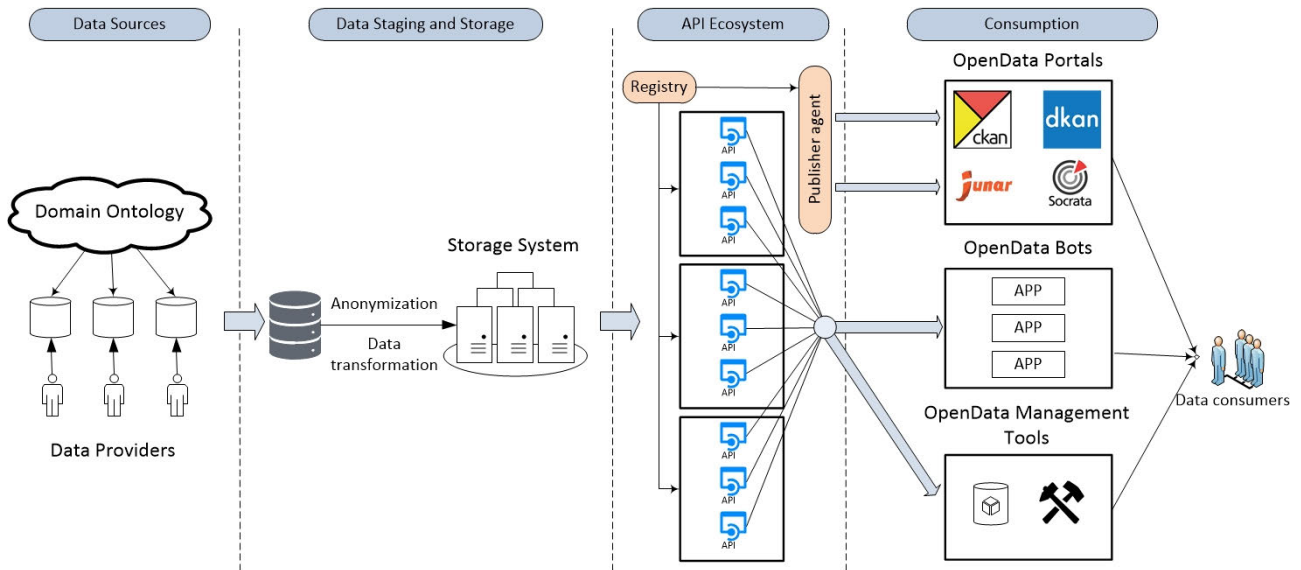


FIGURE 2. Sustainable open data management system architecture.

C. REQUIREMENT 3: USABLE OGD

The provision of new digital channels of interaction based on natural language processing by conversational bots should be addressed as a key enabler to foster OGD usability and then, OGD consumption for end-users. The main aim is to create a user interaction that is as natural as possible for querying and retrieving OGD.

Developing conversational bots for OGD involves addressing two essential issues. On the one hand, interpreting the natural language adopted by end-users to handle the conversation in querying datasets and, on the other hand, translating it into effective queries to build up the answer from datasets.

V. ARCHITECTURE

The system architecture we propose (see Figure 2) consists of four components: 1) data sources, 2) data staging and storage, 3) API ecosystem, and 4) consumption.

The two core components proposed for the system architecture are the API ecosystem to ensure long-term publishing considering the timely and accessible data requirements. Consequently, it is proposed to publish/unpublish OGD in an automated way. Second, the consumption component must ensure long-term consumption considering the usable data requirement.

The data sources component represents a unified representation based on an ontological approach [44]–[46] for different heterogeneous data gathered from diverse sources. This component provides a schema for the semantic representation of data sources, which is consistent and complete enough to model all the aspects and entries for datasets. The data staging and storage component is responsible for data storage and implementing the anonymization policy in such a way that identifiable personal characteristics are hidden before datasets are managed by the API ecosystem component.

In each of the following subsections, we formalize, describe and provide the novel aspects of the core components of the architecture.

A. API ECOSYSTEM

The “ecosystem” metaphor is traditionally used in open data scenarios to describe a set of interdependent elements that together form an evolving, self-organizing and sustainable system. Various works are found within the research community with different viewpoints. Among them are those proposing the basic components and definitions from a business model perspective [47], defining the kind of knowledge that is required for validating open data in digital service ecosystems [48], evaluating the benefits of open data [49], analyzing the major stakeholders [50], and national experiences focused on social, political, and economic angles [51]. Our approach differs from them in two essential aspects. Firstly, our approach is directly related to an evolvable and scalable ecosystem supported on API resources for dynamically creating, managing, and sustaining these resources to provide timely dynamic data, which is not covered by existing data ecosystem approaches. Secondly, we propose to automate the governance of the API ecosystem to ensure the sustainability of the data management system by a software agent, which is also not covered by existing data ecosystem approaches.

Formally, the API ecosystem is defined in terms of a triplet, denoted by $E = (Re, A, RI)$, where E is the API ecosystem name; Re is the set of API resources produced and consumed by ecosystem actors; A is the set of actors who participate in the ecosystem; and RI is the set of relationships engaged by ecosystem actors.

The environment of the ecosystem evolves through a set S of discrete states, where $S = \{s_0, \dots, s_n\}$. Therefore, given an ecosystem environment E , we write henceforward

$E(s_t)$, to denote the ecosystem is at a finite state s_t ; and $Re_i(s_t)$, $A_i(s_t)$ and $Rl_i(s_t)$ to denote respectively a specific API resource i , actor i or relationship i at a state s_t .

In the following, the formalization and description of each one of these conceptual elements, the proposal of the design of API resources based on a REST architectural style (henceforward API resources), and the approach on how to automate the governance of the ecosystem by a software agent is presented.

1) API RESOURCE

Given an ecosystem environment state s_t , an API resource is formalized as an octuple denoted by $Re_i(s_t) = (\{Ca_1, \dots, Ca_k\}, \{En_j\}, \{Pt_1, \dots, Pt_q\}, \{Nt_j\}, \{At_1, \dots, At_m\}, \{St_1, \dots, St_n\}, \{Ql_1, \dots, Ql_k\}, \{Li_1, \dots, Li_l\},)$, where:

- Re_i is the API resource name.
- $\{Ca_1, \dots, Ca_k\}$ refers to a set of categories. Each category Ca_i is established by the reference ontology of the domain. Each resource Re_i is grouped into a specific category Ca_i . An additional category, called *API management*, is included to handle the different aspects related to ensuring the automated governance of API resources by the software agent.
- $\{En_j\}$ refers to the location of the path item to the endpoint for each API resource Re_i .
- $\{Pt_1, \dots, Pt_q\}$ refers to a list of ODPs on which resource Re_i has been published. In the case the list is empty, resource Re_i has not been published on any data platform.
- $\{Nt_j\}$ represents the communication channel of the ecosystem's actors to notify the results of a resource Re_i in terms of its quality properties. For instance, the email address that will be used to notify ecosystem's API producers of the outcomes of a resource Re_i in terms of its quality properties, that is, its reliability and performance.
- $\{At_1, \dots, At_m\}$ refers to a set of two attributes that characterizes the features related to the accessibility of each resource Re_i . The first one identifies the level of access for each API resource. The second attribute identifies the type of resource according to the way the URL patterns have been constructed. The values for the first attribute are *public* or *private*. Public APIs are accessible by all data consumers. These APIs are *read-only* APIs (i.e., only GET operations are considered). Private APIs are internal APIs only accessible to the software agent to facilitate core functionalities for creating and managing APIs resources and, are *updateable* APIs (i.e., all operations are considered). The values of the second attribute are *fixed* or *variable* to indicate if it is a concrete or varying resource (see Section V-A2).
- $\{St_1, \dots, St_n\}$ is a set of standards. Each standard St_i represents the specification to which the described resource Re_i is conformed. The Open API Specification [22] is the specification to define the resources

and operations. It allows both human and computers to understand the resources and operations of APIs without looking at the source code and extra documentation.

- $\{Ql_1, \dots, Ql_k\}$ refers to a set of four quality properties to which each API resource Re_i is evaluated. The first one, Ql_1 , is related to testing outcomes. These include the results for both nominal and faulty specification-based test cases. Nominal tests assess that given correct input data, the API resource operations return a successful response code (i.e., 2xx family of codes). Faulty tests assess that given incorrect input data, the resource operations return an error response code (i.e., 4xx or 5xx family of codes) [52]. As a result of these testing outcomes, for each ecosystem environment state s_t , the value of the first quality property, Ql_1 , is computed according to the following decision rule:

$$Ql_1(s_t) = \begin{cases} 1, & \text{if nominal and faulty} \\ & \text{testing are passed} \\ 0, & \text{Otherwise} \end{cases} \quad (1)$$

The second quality property, Ql_2 , refers to the number of times each resource Re_i has passed both tests. Accordingly, this property is associated with the *reliability* of each resource representing how "effective" the resource is in terms of robustness: the higher the reliability, the better. For each ecosystem environment state s_t , the value of Ql_2 is computed according to the following expression:

$$Ql_2(s_t) = \sum_{i=0}^n Ql_1(s_{t-i}) \quad (2)$$

This gives a reliability measure for each resource considering past testing results and not only the current result, where n represents the dimension of the subset of previous environment states to be considered. Reliability increases as the number of nominal and faulty tests are passed in the recent past. The third quality property, Ql_3 is related to the performance of each resource Re_i in terms of its response time (i.e., the time to react to a request once it has received one). The fourth quality property, Ql_4 , provides a measure of the quality of code written for each resource Re_i in terms of statement coverage testing. It describes the degree to which the statement code of each resource is executed when test cases are executed, verifying what the written code is expected to do and not do. A resource with high statement coverage, measured as a percentage, suggests it has a lower chance of containing undetected software bugs compared to resources with low statement coverage.

- $\{Li_1, \dots, Li_l\}$ is a set of licenses. Each license Li_i represents a license to perform some activity related to resource Re_i . The open license Open Data Commons Public Domain Dedication and License (PDDL) can be

selected to freely share, modify, and use the datasets for any purpose and without any restrictions.

2) BUILDING API RESOURCES

The design of API resources must focus on how to automatically publish dynamic data and datasets via APIs on ODPs, when new data and datasets are made available by organizations. Two types of API resources called *variable* and *fixed* were proposed by analyzing data models for storage and retrieval of data produced in organizations which are essentially dynamic considering they evolve over time according to the performance of their main functions. With variable API resources, we refer to a structured way for defining how the URL patterns should be constructed to allow the creation of new datasets which are not known a priori (e.g., datasets on new parliament members and laws of the next legislature). This is distinct from a fixed API that specifies a concrete resource name which is unique over time and different legislatures (e.g., a dataset on municipalities or a dataset on assets). In this case, the syntax of the URL pattern is structured as follows:

```
//SITE/API_URL
```

where *SITE* represents the domain of the site, and *API_URL* represents the API path. An example of a URL pattern of a fixed API resource for retrieving a dataset corresponding to the vehicles owned by a parliamentary institution is the following:

```
//parcan.es/api/transparency/vehicles/
```

where *parcan* refers to the site of the parliamentary institution, *transparency* corresponds to the API category Ca_i , and *vehicles* refers to the resource name.

On the other hand, when we are designing variable API resources, we provide a dynamic way for building specific resources, where it is assumed that the values of path parameters (e.g., members, legislatures, legislative bodies, years, etc.) that specify variable collections for each different API are not known ahead of time and evolve over time.

With this aim, the path templating approach [53] is used as a mechanism to specify how to describe sets of relative URL patterns when variable values need to be provided according to the evolution of the data over time and are not known a priori. The syntax of the URL pattern for the proposed variable API resources is then structured as follows:

```
//SITE/API_URL/PARAMETER
```

where *SITE* represents the domain of the site, *API_URL* represents the API path, and *PARAMETER* represents the parameter element. Curly brackets are used to denote parts of the parameter element as a path variable, which is not known ahead of time. An example of a URL pattern of variable API resources for retrieving datasets corresponding to the agenda of members by year and months in a parliamentary setting,

where the data values corresponding to them are not known in advance is the following:

```
//parcan.es/api/activity/deputies/agenda  
/{member}/{year}/{month}
```

3) DYNAMIC GENERATION OF API RESOURCES

Fixed API resources are automatically published on ODPs since their URL patterns directly represent the URL path that data consumers must invoke to obtain the required datasets. In the case of variable API resources, a four-step process is developed to ensure the dynamic generation of all specific resources for variable collections where it is assumed that the path parameters and their values are not known ahead of time and evolve over time (e.g., year, code of a legislature, parliamentary member, etc.). This process is executed by the software agent integrated into the API ecosystem. Three essential API resources grouped into the *API management category* called *API query*, *API requests resolver* and *API generator* must be developed. The four-step process is as follows:

- The *API query* first builds the query according to the following format to obtain each of the different API categories Ca_i :

```
https://SITE_NAME/api/request_resolver/?category=CATEGORY_VALUE
```
- The different requests to the *API requests resolver* with the details about the required categories Ca_i are performed in the second step of the process to obtain the set of all API resources Re_i .
- In the third step, the *API requests resolver* returns the values for each parameter expression of each variable API resource Re_i that correspond to each representative API category Ca_i .
- In the last step of the process, the *API generator* is responsible for processing the inferred values from the expression expansion for each URL template variable by generating as many API resources Re_i as possible values the parameters named within the expression have. Accordingly, the returned API resources Re_i consist of a set of URL templates with their values as defined by the corresponding parameter types.

As a result of this four-step process, each possible resulting URL pattern is obtained for all the different API categories Ca_i . Figure 3 illustrates an excerpt with the outcomes obtained in JSON format in a parliamentary setting, when the *API requests resolver* is requested by the specific API category representation and after the first three steps of the process have been performed. The results shown in the Figure 3 include the different API resources obtained, the URL template variable, and the values required for each path parameter needed (l_{island} , $l_{legislature}$). Subsequently, in the last step of the process, all the URLs with their specific resources and each inferred parameter value will be generated by the *API generator* for each of the seven different legislatures and seven islands.

```

{"Api_Category": "representation",
 "urls": [
 {"api": "representation_parliamentary_bodies_legislature",
  "url": "https://parcan.es/api/representation/parliamentary_bodies/legislature/{1_legislature}/"},
 {"api": "representation_deputies_biography",
  "url": "https://parcan.es/api/representation/deputies/biography/{1_legislature}/"},
 {"api": "representation_deputies_legislature",
  "url": "https://parcan.es/api/representation/deputies/groups/{1_legislature}/"},
 {"api": "representation_deputies_island",
  "url": "https://parcan.es/api/representation/deputies/island/{1_island}/"},
 {"api": "representation_bureau",
  "url": "https://parcan.es/api/representation/bureau/{1_legislature}/"},
 {"api": "representation_speakers_board",
  "url": "https://parcan.es/api/representation/speakers_board/{1_legislature}/"},
 {"api": "representation_plenary_session",
  "url": "https://parcan.es/api/representation/plenary_session/{1_legislature}/"},
 {"api": "standing_commissions",
  "url": "https://parcan.es/api/representation/parliamentary_bodies/commissions/standing/{1_legislature}/"},
 {"api": "study_commission",
  "url": "https://parcan.es/api/representation/parliamentary_bodies/commissions/study/{1_legislature}/"},
 {"api": "ponencias",
  "url": "https://parcan.es/api/representation/parliamentary_bodies/presentations/{1_legislature}/"},
 {"api": "subcommissions",
  "url": "https://parcan.es/api/representation/parliamentary_bodies/subcommissions/{1_legislature}/"}
 ],
 "1_island": ["la_palma", "el_hierro", "la_gomera", "tenerife", "gran_canaria", "fuerteventura", "lanzarote"],
 "1_legislature": [10, 9, 8, 7, 6, 5, 4]
}

```

FIGURE 3. Excerpt of the results of the API requests resolver for the API category representation in JSON format.

4) AUTOMATED GOVERNANCE

Automated governance of all API resources in the API ecosystem is addressed by a software agent with internal state (henceforward called API Publisher, Publisher Agent or agent). The Publisher Agent runs continuously in the API ecosystem to perform three functions: perception of the dynamic conditions in the ecosystem, updating the internal state with new perceptions, and actions upon that ecosystem to carry out its goals. The agent decision-making is modelled considering the ecosystem and its evolution and not solely based on the present. The agent is modelled assuming that the environment of the ecosystem may be in any state of a set S of discrete states, and it has a repertoire of possible actions A to transform the state of the environment, where $A = \{a_0, \dots, a_m\}$. Formally, the behavior of the state-based agent can be represented by the following functions [39], [54]:

See, which captures the agent's ability to observe its environment. The output of the function is a *percept* (i.e., a perceptual information). Let *Per* be a (non-empty) set of percepts, the function *see* is a function which maps environment states to percepts:

$$see : S \rightarrow Per$$

Next, which captures the ability of the agent to record information about the environment state and history. Let I be the set of all internal states of the agent, the function *next* maps an internal state and percept to an internal state:

$$Next : I \times Per \rightarrow I$$

Action, is defined as mapping internal states to actions upon the environment. Thus, the agent decides about what action to perform based on the evolution of the environment and not solely based on the present.

$$Action : I \rightarrow A$$

The ecosystem is supported by a software environment. The different elements of the Publisher Agent, the software environment and behavior for the automated governance of API resources is illustrated graphically in Figure 4. The environment comprises an API Registry, an API Repository and Open Data Platforms. The API Registry provides a registration mechanism for advertising all available API resources. The API Repository provides access to the software implementation of different API resources. The Open Data Platforms are the data infrastructures on which API resources are published/unpublished.

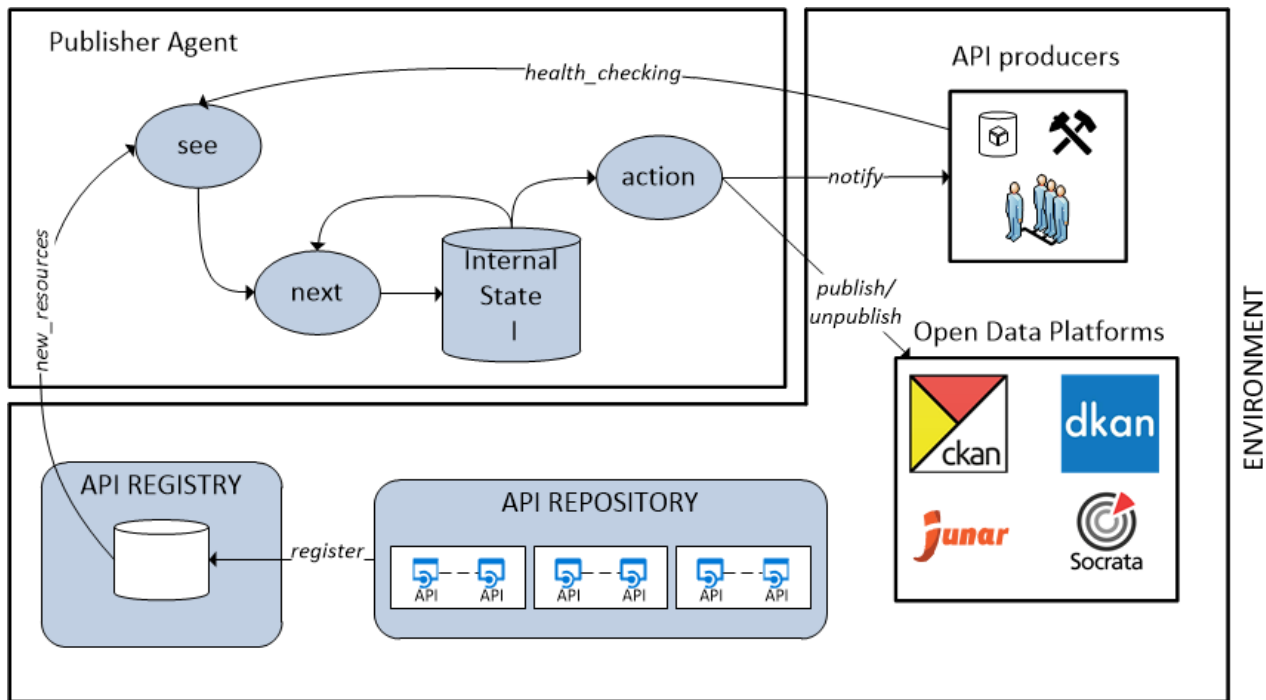


FIGURE 4. The Publisher Agent in its environment for the automated governance of API resources.

The Publisher Agent can receive two different percepts: a percept *new-resources* (a trigger from the API Registry, indicating that new datasets are available to be published as API resources) or *health-checking* (a trigger from ecosystem’s API producers, signifying that the enabled-API resources need to be validated for detecting reliable/faulty resources and, subsequently enabling their publishing/unpublishing on ODPs). As mentioned in Section V-A2, the API resources produced can be fixed or variable.

The internal state of the Publisher Agent represents the information the agent has about the set of available API resources. It varies over time according to new percepts and the environment’s evolution. The information the agent has about its environment for each available resource Re_i is name, category, location, open data platforms, notifications, attributes, standard, quality, and license (see Section V-A1). The set of all internal states of the agent is represented by I . A specific internal state of the agent is then an element of I , where $\Delta_1, \dots, \Delta_n$ denotes the different members of I .

The perception function *see* maps the environment states to two different percepts (*new-resources*, *health-checking*). The *next* function is specified according to the perceptual information obtained from the environment (either *new-resources* or *health-checking*) and generates a new set of internal states I , which includes this information. But, in addition, the agent must also remove old or irrelevant information. Therefore, the *next* function is specified in two parts. First, let *old* be (Δ) the set of old information about the set of all API resources in I to remove. Then, let the function *new* be the set of new information about API resources to add

to I :

$$new : I \times Per \rightarrow I$$

The function *new* of the agent updates the set of internal states I according to the two different percepts that it can receive. When the percept is *new-resources*, for each new resource Re_i produced, the function *new* updates the set of internal states I by (1) carrying out the dynamic generation of all possible API resources (see Section V-A3) in the case of a variable API or adding the API resource in the case of a fixed API. And subsequently, (2) storing resource Re_i and adding the information corresponding to each of the eight components that characterizes each API resource obtained.

When the percept is *health-checking*, the function *new* validates the set of API resources of the agent’s current internal state in terms of reliability and performance. With this aim, a testing process that consists of both nominal and faulty specification-based test cases, as well as performance testing is performed for each resource Re_i . As a result of this process, the values of each of the four quality properties (Ql_1, Ql_2, Ql_3 and Ql_4) are obtained for each resource (see Section V-A1) and updated in the current internal state of the agent.

Given the *new* and *old* functions, the *next* function of the agent is defined as follows:

$$next(I) = (I \setminus old(\Delta)) \cup new(I)$$

The decision-making of the agent on what action to carry out on its environment is influenced by the ecosystem’s history, and it is modelled by behavior rules regarding publishing reliable API resources on ODPs, unpublishing

faulty API resources on ODPs, and notifying faulty API resources to ecosystem's API producers. The goal is to ensure automated publishing, testing and removal of API resources over time. Consequently, publishing must guarantee the automated publishing of new API resources and their timely updates on data platforms over time when new datasets are available, testing must ensure the proper operation of all API resources, whilst removal must remove API resources when datasets are no longer available, or API resources are faulty.

The first rule that governs the agent's behavior, a_1 , deals with the publishing action. Since the quality properties, Ql_2 and Ql_3 , represent a measure of effectiveness and performance for each resource Re_i considering past testing results, the agent can decide when performing a publishing action is appropriate or not, focused on this measurement. Hence, for a given environment state s_t , if the agent detects that the quality property Ql_2 (i.e., the reliability) of a resource Re_i is higher than a publishing threshold, $\delta_{publishing}$, and the response time of the resource is lower than a response time threshold $\gamma_{response}$ and, the resource has not been published on a data platform Pt_i , then the prescribed action will be to publish the resource Re_i on the corresponding platform Pt_i :

$$\begin{aligned} & \text{if } (Ql_2(s_t) > \delta_{publishing}) \quad \text{and} \\ & \quad (Ql_3(s_t) < \gamma_{response}) \quad \text{and} \\ & \quad (Pt_i(s_t) = 0) \quad \text{then} \\ & \quad \text{publish } Re_i(s_t) \text{ on } Pt_i(s_t) \end{aligned} \quad (3)$$

The second rule, a_2 , deals with the unpublishing action of the agent. Hence, for a given environment state s_t , if the agent detects that the quality property Ql_2 for a resource Re_i is less than or equal to a publishing threshold, $\delta_{publishing}$, and it has been published on a data platform Pt_i , then the prescribed action will be to unpublish the resource Re_i on the corresponding platform Pt_i :

$$\begin{aligned} & \text{if } (Ql_2(s_t) \leq \delta_{publishing}) \quad \text{and} \\ & \quad (Pt_i(s_t) = 1) \\ & \quad \text{then unpublish } Re_i(s_t) \text{ on } Pt_i(s_t) \end{aligned} \quad (4)$$

As a result, the agent's decision-making to publish/unpublish a resource Re_i is based on the temporal coherence of the testing results by considering the outcomes obtained through the recent history and not just the current outcome. This is performed by considering that each testing process for a resource is required to be passed at least $\delta_{publishing}$ times for a publishing action. Whilst for an unpublishing action, it is required that the resource has at least $n - \delta_{publishing}$ testing failures, where n represents the number of previous environment states to be considered. This guarantees that a resource will not be immediately published/unpublished by the agent when a testing process is passed or failed.

The third rule, a_3 , deals with the notification action about faulty resources to ecosystem's API producers. Hence, if the agent detects that a resource was published in a previous state

s_{t-1} and in the current state s_t is unpublished then the action will be to notify the faulty and unpublished resource to the API producers:

$$\begin{aligned} & \text{if } (Pt_i(s_{t-1}) = 1) \quad \text{and} \\ & \quad (Pt_i(s_t) = 0) \quad \text{then} \\ & \quad \text{notify unpublished } Re_i(s_t) \text{ to } Nt_i(s_t) \end{aligned} \quad (5)$$

The fourth rule, a_4 , refers to the notification to ecosystem's API producers when the performance in terms of response time does not exceed a response time threshold. Thus, the action will be to inform API producers about the faulty resource:

$$\begin{aligned} & \text{if } (Ql_3(s_t) > \gamma_{response}) \quad \text{then} \\ & \quad \text{notify faulty resource } Re_i(s_t) \text{ to } Nt_i(s_t) \end{aligned} \quad (6)$$

As a result of the interaction of the Publisher Agent with its environment, its behavior for the automated governance of API resources can be summarized in the following way. The agent starts in the API ecosystem in some initial internal state i_0 . It then observes its environment state s_t , and receives a percept $see(s_t)$, either *new-resources* or *health-checking*. The internal state I of the agent is then updated via the *next* function, becoming set to *next* ($i_0, see(s_t)$), removing old information and adding new information to the corresponding resources Re_i . The action selected by the agent is then *action(next($i_0, see(s_t)$))* related to publishing, unpublishing and notifying actions. The corresponding action is then performed, and the agent enters another cycle, perceiving the ecosystem via *see*, updating its state via *next*, and choosing an action to perform via *action*. Each cycle of the Publisher Agent is defined considering the required frequency to provide timely datasets.

5) ACTORS

Actors are essential elements of the API ecosystem and each one has different roles and capabilities. Given an ecosystem environment state s_t , an actor is denoted by $A_i(s_t) = (\{Ro_1, \dots, Ro_l\}, \{Re_1, \dots, Re_n\})$, where:

- A_i is the actor name. The actors of the ecosystem are integrated both by humans and software agents. Three types of actors are defined according to the role played in the API ecosystem: API producers, API publisher or API consumers.
- $\{Ro_1, \dots, Ro_l\}$ is a set of roles. Each role Ro_i represents a role performed by actor A_i . The role of API producers is to manually develop fixed API resources. API publisher is an autonomous software agent, whose role is to ensure the dynamic generation of variable API resources and a sustainable governance of all of them to guarantee the proper operation, as well as timely publishing/unpublishing of all resources. API consumers are both developers and end-users. Developers access the API resources produced in the ecosystem to generate data-enabled products and innovation. Potential end-users in parliamentary settings include citizens,

public administration, government, local authorities, media, researchers, parliamentary staff, lawyers, and practitioners.

- $\{Re_1, \dots, Re_n\}$ is the API set of resources. Each resource Re_i represents an API resource produced or consumed by actor A_i .

6) RELATIONSHIPS

Relationships are the interactions between API ecosystem actors. Since open license is selected to freely share, modify, and use datasets for any purpose and without any restrictions, no relationship follows a business model. Given an ecosystem environment state s_t , a relationship Rl_i is formalized by $Rl_i(s_t) = (\langle A_i, A_j \rangle, \{At_1, \dots, At_n\})$, where:

- $\langle A_i, A_j \rangle$ is a pair of actors, which represents the actors that participate in relationship Rl_i .
- $\{At_1, \dots, At_n\}$ is a set of attributes. Each attribute At_i represents a feature that characterizes relationship Rl_i .

B. CONSUMPTION

OGD is published as API resources on ODPs for data consumers, as illustrated in Figure 2. The publication of OGD as API resources facilitates the use of bots as conversational interfaces for querying published data and the use of data management tools to develop machine learning services to offer functionalities such as data mining, classification, and data interpretation. The following subsection addresses how an OGD API-driven bot can be used as a conversational bot to enable OGD usability considering the Google Dialogflow platform.¹⁰

1) OGD API-DRIVEN BOT

The aim of the resulting OGD API-driven bot (henceforward called bot) is to provide a user-friendly access and querying method to datasets as well as the ability to help end-users to gain knowledge from the datasets, which represent a domain environment and create new knowledge from them.

The development of the bot poses different challenges, such as interpreting the natural language adopted by end-users to handle the conversation in querying the information needs on OGD and, translating it into effective queries for the resulting API ecosystem to build up an answer. To enable the bot to provide such functionalities, two components need be addressed: the knowledge base (to provide responses to end-user input) and the dialogue management module (to handle the conversation process).

For the development of the knowledge base, the ontological perspective of the domain, which formally specifies the conceptual terms and semantic relationships that model and represent the domain is used. As a result, the bot system's knowledge base is obtained from the semantic model of the domain and then, it is internally represented according to the different categories in the API resources that encompass the API ecosystem. Bot's knowledge base is updated over

time as the API ecosystem is updated dynamically by the Publisher Agent (see Section V-A4) when new datasets are made available.

For the development of the dialogue management, Dialogflow is used to (1) understand end-users input queries expressed in natural language processing (NLP) by speaking or typing a question, (2) generate a dynamic response from the API ecosystem to fulfill the request and, (3) present a response to end-users according to a response template.

To build bot's behavior with Dialogflow, the following components are defined: intents, entities, contexts and fulfillment. An intent represents the intention behind end-users' inputs and the goal expected to be achieved with each request. Consequently, one intent is modelled for each type of end-user request that the bot can support. Dialogflow contexts are used to control the flow of a conversation for an intent by setting input and output contexts. Dialogflow fulfillment is used to provide a dynamic response according to the bot's knowledge base instead of static responses for matched intents. With this aim, webhook integration [55] is used in terms of requests to the bot's webhook services to handle dynamic responses by passing information from a matched intent into the corresponding API calls to the API ecosystem and getting the result from it. This guarantees that the bot will get the appropriate answer when an end-user issues a question in a parliamentary setting like "*Will the bill on "Assistance dogs for people with disabilities" be voted on in the next plenary session?"*". This is because it performs the actions as calls to API resources of the API ecosystem, which is updated timely by the Publisher Agent. Accordingly, the bot will always give available information considering the updated parliamentary agenda corresponding to that session instead of static responses of matched intents. This is essential since static responses of matched intents cannot include information which is dynamically updated based on the parliamentary activity that is going to occur over time.

The user interface of the bot can be implemented as a mobile app, where voice and text can be supported as modes of communication. The bot's behavior is modelled by the following seven-step process as follows:

- Step 1) The end-user interacts through the interface by speaking or typing a question.
- Step 2) Dialogflow matches the end-user expression to an intent and extracts parameters.
- Step 3) If an intent is matched, a webhook request message with information about the matched intent is sent by Dialogflow to the bot webhook service.
- Step 4) The bot webhook service performs the actions as API calls to the API ecosystem.
- Step 5) The bot webhook service sends a webhook response message to Dialogflow. This message contains the response to be sent to the end-user and updates to the context active for the conversation.
- Step 6) Dialogflow sends the response to the end-user.
- Step 7) The end-users hear or see the response.

¹⁰<https://cloud.google.com/dialogflow>

VI. EVALUATION

In this section, we evaluate the novel aspects of the two core components of the architecture proposed in section V considering the requirements for these components (specified in Section IV) when the data management system is implemented as a sustainable solution to support the open data initiative for the Parliament of the Canary Islands for long-term OGD publishing and consumption. The API ecosystem component of the architecture is evaluated for long-term OGD publishing, whereas the OGD API-driven bot component is evaluated for long-term OGD consumption.

The entire infrastructure that supports the solution consists of a hyperconverged infrastructure (HCI) with virtualization software with 4 nodes VxRAIL E560 Hybrid + E560F All Flash. Each node has 2xIntel Xeon 56 @ 2.5Ghz (16 cores per CPU) with 384 GB of RAM. As a result, the nodes provide a storage cluster vSAN All Flash of 121 TB. This HCI has been selected as infrastructure to support our solution because it provides a unified system that decreases data center complexity and increases data scalability. The reference ontology of the domain described in our previous research work [56] was developed using the Protégé ontology tool [57] as well as Virtuoso universal server and RDBMS Oracle, which have been used as database engines in the data layer. Django REST framework¹¹ has been used as toolkit to develop the API ecosystem, where the API resources rely on the OpenAPI specification [22]. Currently, CKAN¹² is utilized as ODP to provide access to all available data via API resources.

Which data must be anonymized and which anonymization technique should be used, in such a way that identifiable personal characteristics are hidden when OGD are published and consumed, were defined according to an anonymization policy. This policy was based on the European General Data Protection Regulation [58] that currently is the most relevant existing regulation for privacy protection for all European individuals since May 2018. The attributes that can identify a specific individual (i.e., members of parliament and civil servants) were classified into three categories: (1) key attributes (attributes that uniquely identifies individual, e.g., ID, name, social security number); (2) quasi-identifiers (attributes that can be combined with external information to expose some individuals, e.g., home address); sensitive attributes (attributes that contain sensitive information about individual, e.g., disability status). Generalization and suppression techniques [59] were applied respectively to replace a QI value with a less specific value to reduce the granularity of representation by using the l-diversity model, while suppression is focused on hiding a QI value entirely.

The autonomous agent has been developed as a goal-based agent with Python programming language. The OGD API-driven bot was developed as a prototype of conversational bot with Google DialogFlow platform to enable parliamentary

open data usability. Currently, other platforms are being studied to deploy the bot and its integration with the parliamentary web portal as well as other messaging platforms.

The Verification, Validation and Testing model [60] was adopted to evaluate the sustainable data management system focusing on the two core components proposed for the system architecture: the API ecosystem and the conversational bot. Each process involved in the evaluation is discussed below.

A. VERIFICATION

The verification process established whether the data management system implemented meets the timely data and accessible data requirements for the API ecosystem and the usable data requirement for the OGD API-driven bot as well as the ability of the data management system to be sustainable over the entire evolution cycle. The solution proposed is also compared with existing approaches for both OGD publishing and OGD consumption.

Regarding the framework proposed, SuDaMa, it can be affirmed that it is effective and efficient as a practical solution to support the open data initiative for the Parliament of the Canary Islands. This initiative started on January 2019, by carrying out the methodology described in Section III and providing the resulting data management system. This was affirmed by the IT CIO of the Parliament of the Canary Islands, the project advisory board, parliamentary stakeholders and the ULL. As evidence of this, the deployed solution remains operational and has been continuously validated by the Parliament with over 3,000 datasets published by the API ecosystem after two years since commissioning. The data have been verified and timely provided and are accessible in a user-friendly way via APIs to developers as well as enhancing the usability of data by end-users.

Regarding the framework proposed, SuDaMa, in comparison with the software approaches described in the literature for implementing OGD publishing solutions (see Section II-B), it can be stated that the solution outperforms the affordances of the three existing approaches. First, when we compare the solution proposed against an approach focused on using existing ODPs (i.e., publishing directly on ODPs), our solution is more efficient because our solution not only allows data publishers to publish data on any ODP with data maintenance over time but also ensures real-time access to dynamic data as well as generating dynamically new datasets when they are available. Second, when we compare the solution proposed against the other two approaches focused on developing custom ODPs as well as developing data management systems and their integration with existing ODPs, our solution also performs better than the existing ones since our solution does not use ETL tools to carry out the publishing process. As a result, our solution avoids the weaknesses of these approaches. That is, our solution can ensure real-time publishing of dynamic data without using manual development effort every time a data release is available, which not only eliminates errors, but also

¹¹ <https://www.django-rest-framework.org/>

¹² <https://ckan.org/>

leads to significant cost savings in comparison to traditional approaches focused on the use of ETL tools.

Regarding the framework proposed, SuDaMa, compared to other software approaches described in the literature for implementing OGD consumption solutions (see Section II-C), provides a novel contribution by introducing conversational bots to exploit the potential of OGD and thus improve the consumption experience of end-users in terms of creating an end-user interaction as natural as possible for querying and retrieving OGD.

Moreover, in contrast to previous solutions described in the literature [3], [28]–[36], our proposed solution has been designed making sustainability an explicit consideration of the design of the data management system (i.e., the capability of the system to last over time in an automated way). This not only eliminates development errors in publishing/unpublishing tasks but also lowers development and maintenance costs.

Regarding the main conclusions of the CIO, he reflected the importance of the results achieved. On the one hand, the setting up and use of the API ecosystem has been based on the principles of availability, stability, maintenance over life cycle, uniformity of use and standards and user-friendliness. On the other hand, the system has been verified as being reliable and scalable. That is, the assurance that the system is behaving and responding as intended and that it can scale up according to increasing amounts of datasets published via API resources. The CIO also highlighted that currently, the Canary Islands Parliament is the only one of all seventeen Spanish parliaments that publishes data on ODPs, and, to the best of his knowledge, is the first regional parliament that has implemented a sustainable open data system for publishing and consumption over time as well as introducing a bot as conversational interface for consuming OGD.

The sustainable data management system has also been verified as a solution in practice to support the open data initiative of the Canary Islands Parliament with a demonstration of the solution given to the president of the parliament, media and a wide diversity of stakeholders on 20th June 2019.

B. VALIDATION

The validation process was aimed at evaluating the sustainability and efficiency of the system considering the two core components proposed of the architecture: the API ecosystem and the conversational bot. The validation of the first and second requirement described in Section IV (timely data and accessible data) validates the API ecosystem for long-term OGD publishing, whereas the third requirement (usable data) validates the conversational bot for long-term OGD consumption.

1) VALIDATION OF REQUIREMENT 1: TIMELY OGD

For the first requirement to be met (see Section IV-A), the first core component of the architecture, the API ecosystem should be evolvable and scalable by carrying out the dynamic

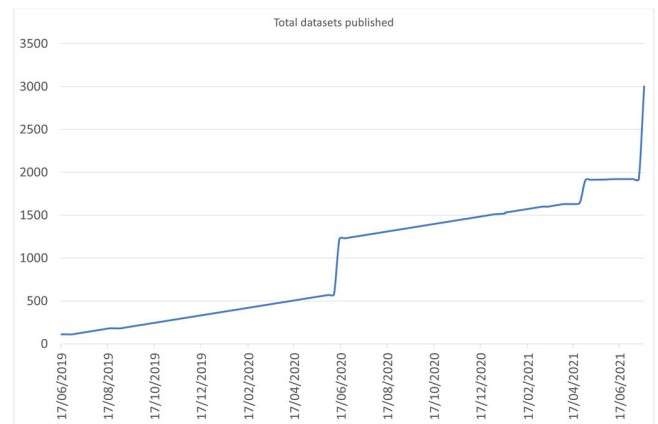


FIGURE 5. Statistics on the publication of new datasets on CKAN as new API resources are generated.

generation of all possible API resources (see Section V-A3) in the case of variable APIs when the Publisher Agent receives a percept *new-resources* (see Section V-A4). This has been validated by checking when new resources have been generated with the correct publication on the open platform CKAN. Figure 5 illustrates the statistics on the publication of new datasets on CKAN as new API resources are generated. The statistics show that the number of datasets has been constantly growing since the system was deployed. This is in line with the evolution of the data according to the work of this parliamentary institution. As result of this validation process, it has been proved that the automation of the generation of API resources behaves correctly to provide up-to-date open data over time.

Each discrete state s_t of the agent represents the execution of an agent's cycle, which is set considering the required frequency to provide timely datasets. Currently, in our parliamentary context, this cycle is once per day. However, this cycle can be increased or decreased according to the need for providing new datasets on ODPs.

2) VALIDATION OF REQUIREMENT 2: ACCESSIBLE OGD

For the second requirement to be met (see Section IV-B), developers must evaluate the adoption of API resources as the main asset of the API ecosystem in terms of easy access to published datasets, therefore, facilitating their implementation in data-enabled products. The assessment was performed from the perspective of the field of technology assessment (TA) [61], evaluating how API technology facilitates the implementations of data-enabled products for developers compared to the traditional retrieval method based on bulk data. In this context, the perceived usefulness and ease of software development are important factors that influence developers' decisions. To do this, we developed a 7-point Likert scale assessment questionnaire, shown in Table 1, with four items with the endpoints *Not at all* (1) and *Yes, totally* (7) with ten developers. Figure 6 shows the results of the questionnaire for each question from the distinct participants. For all questions, high average values were

TABLE 1. Questionnaire employed for the assessment of adopting API resources as technology by developers for implementing data-enabled products.

| Item | Question |
|------|--|
| Q1 | Considering the data accessibility requirement and compared to the traditional bulk data retrieval method from data catalogues, using parliamentary API resources for implementing your data-enabled products is valuable enough? |
| Q2 | Considering the data accessibility requirement, is it sufficiently useful to continue using the parliamentary API resources to implement data-enabled products instead of the traditional bulk data retrieval method from data catalogues? |
| Q3 | Is it sufficiently easy to continue using the parliamentary API resources to implement data-enabled products instead of the traditional bulk data retrieval method from data catalogues? |
| Q4 | Were you able to successfully fulfill your software development with the available parliamentary API resources? |

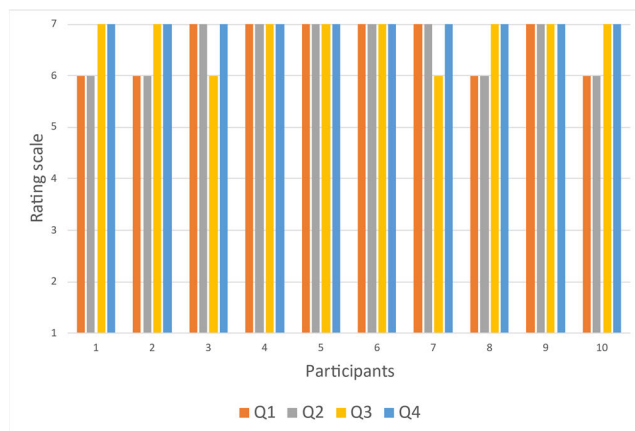


FIGURE 6. Results of the questionnaire by participants and questions, with Q1, Q2, Q3 and Q4 representing Question 1, Question 2, Question 3 and Question 4, respectively.

obtained. Specifically, an average value of 6.6 for question 1, 6.6 for question 2, 6.8 for question 3 and 7 for question 4 were obtained. We can observe that the participants' ratings did not vary and range specifically between 6 and 7. Concerning the ease of software development, the results revealed that all participants were able to successfully implement their applications with the available API resources while 8/10 participants highlighted that the adoption of API resources is considered easier than the bulk data method to implement their data product developments. Regarding the perceived usefulness of adopting API technology for their applications, all participants were also very satisfied, with 6/10 participants rating this aspect with the maximum value of 7 and the remaining 4/10 participants rated it with the second highest value of 6.

Additionally, observational findings on potential benefits and future improvements were obtained from developers' free comments.

After reviewing the feedback, we acquired the following insights on potential benefits: 1) adopting APIs as access method to dynamic data is crucial for developers to

implement real-time based applications as well as providing innovative products that draw on machine learning functionalities; 2) API technology is an appropriate access method compared to bulk data when datasets are too large or so volatile that downloading them all and assuring that all of them are up to date becomes burdensome. Therefore, accessing small parts of data by APIs can lower the barrier to entry and make it easier for developers to begin using the data for their applications; 3) bulk data is static but using APIs is dynamic. As a result, using APIs is expected to have faster response times and higher availability, which means that the service/application runs fast at all times; 4) using APIs is timelier than bulk data since APIs remain continuously and indefinitely available, while bulk data are released once and updated as needed; and 5) the public sector should prioritize the use of APIs to streamline the data economy.

Concerning possible enhancements, two important ones were suggested by participants: 1) providing an online mechanism on the open data portal with a suggestion form according to a demand-driven approach on new datasets could help foster public engagement, new and improved services for citizens, better policy-making processes, advance legislative science, creation of new insights, and generate business value; and 2) APIs should be considered a key enabler not only to access open data but also to deliver legislative digital transformation goals.

3) VALIDATION OF REQUIREMENT 3: USABLE OGD

For the third requirement to be met (see Section IV-C), a prototype of an OGD API-driven bot, called *parcanbot*, was developed using the Google Dialogflow platform to be validated as an application scenario of conversational bot. The aim of the resulting bot system, *parcanbot*, was to provide a user-friendly access and querying method to datasets to increase the transparency and accountability of parliamentary activity as well as the ability to help end-users to gain knowledge from the parliamentary environment and create new knowledge from it. The development of *parcanbot* posed different challenges, such as interpreting the natural language adopted by end-users to handle the conversation in querying the information needs on open data about transparency, parliamentary activity, representation and legislative process and, translating it into effective queries for the resulting API ecosystem to build up an answer. For the development of the knowledge base, our approach built on the ontological perspective of our domain [56], which formally specifies the conceptual terms and semantic relationships that model and represent the main functions of an open parliament. The user interface of *parcanbot* was implemented as a mobile app, where voice and text are supported as modes of communication.

Figure 7 shows the interaction with *parcanbot*. Initially, the end-user accesses the conversational interface of *parcanbot*, which is owned by the Parliament of Canary Islands. *Parcanbot* starts by greeting the end-user. As this is the first interaction of the end-user, the dialogue manager

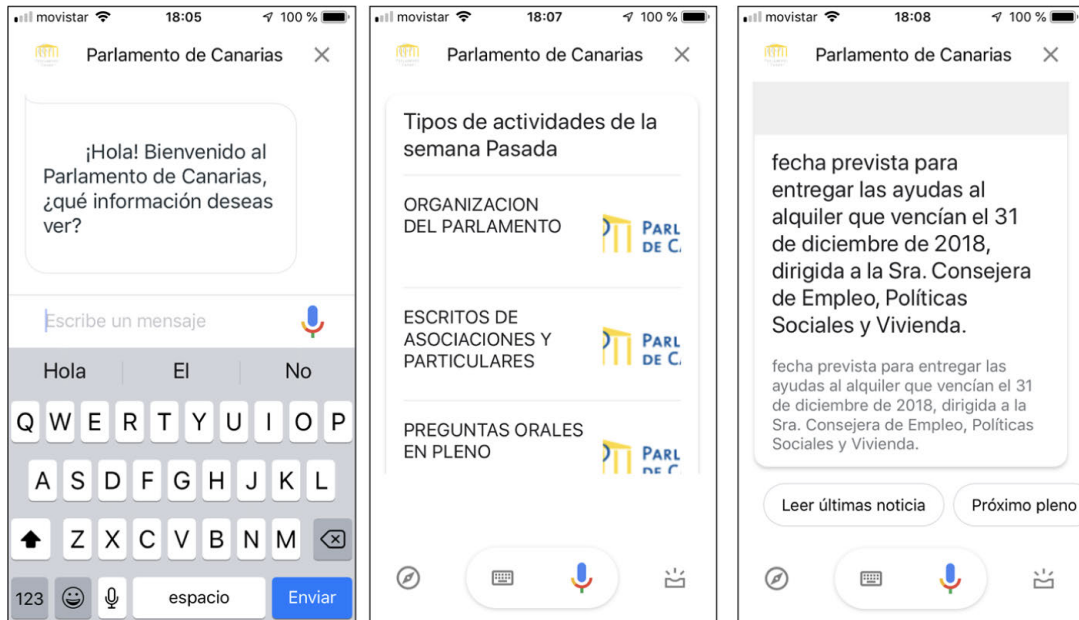


FIGURE 7. Screenshots with instances of the interaction between *parcanbot* and end-user. The language is Spanish in the current prototype.

provides the different clickable options to the interface that *parcanbot* can answer. The end-user can interact by pressing an option or interacting by voice as communication mode. *Parcanbot's* duties include providing information on the procedurally driven dynamics in plenary sessions on what happened (which legislative initiative has been debated and results of any procedural actions), transparency information on representative members and parliament (e.g., agenda, legislative initiatives presented, interventions, declaration of assets) as well as information on parliamentary news and both past and future agendas. In the screenshot shown in Figure 7, the end-user raised a question on parliamentary activity corresponding to two years ago. Accordingly, *parcanbot* responds with different activities corresponding to that period. From now on, the end-user can continue the conversation as he/she prefers, by asking only about the activities conducted in that period: changes related to the organization and operation of the parliament, writings of associations and individuals as well as oral questions in the plenary session. In this case, information related to housing benefits is provided, because the end-user is querying about available oral questions on this topic.

To validate OGD usability using the application scenario of the consumption component, *parcanbot*, a methodology that validates the end-user experience (UX) was used.

The methodology to evaluate UX is based on both a quantitative and qualitative component to validate if the conversational bot is better or worse than the traditional solution based on navigating through an open data portal. For the quantitative evaluation, a suitable User Experience Questionnaire (UEQ) [62] was used as evaluation tool to ascertain the impact that the conversational bot, *parcanbot*, has on end-users, and this result is compared to the overall

impression of using the traditional interface built on the top of the open data portal CKAN for access and navigation. The qualitative evaluation consists of observational findings from end-users about comparing both interface tools and collecting their feedback for future improvement of the bot.

The UEQ provides a quantitative measurement of a product user's experience by allowing end-users to express feelings, impressions and attitudes that arise when experiencing the product in a simple way by filling out a quick questionnaire [62]–[64]. The use of this UEQ has proven to be a valuable tool for comparing technological solutions in real-world scenarios [65]. Specifically, the UEQ contains 6 scales with 26 items, where each item is represented by two terms with opposite meanings. These items are rated on a 7-point Likert scale with the endpoints *Not important at all* (1) and *Very important* (7). Table 2 shows the UEQ provided to the end-users for their assessment, where each item is classified according to one of the following six scales [63]:

- *Attractiveness*: Overall impression of *parcanbot*. Do end-users like or dislike it?
- *Perspicuity*: Is it easy to get familiar with *parcanbot*? Is it easy to learn how to use *parcanbot*?
- *Efficiency*: Can end-users solve their tasks without unnecessary effort using *parcanbot*? Is the interaction with *parcanbot* efficient and fast? Does *parcanbot* react fast to user input?
- *Dependability*: Does the end-user feel in control of the interaction?
- *Stimulation*: Is *parcanbot* exciting and motivating to use?
- *Novelty*: Is *parcanbot* innovative and creative? Does *parcanbot* catch the interest of end-users?

TABLE 2. UEQ employed for the assessment of user experience from [50], where A, P, E, D, S and N denotes respectively the scales associated with each item corresponding to attractiveness, perspicuity, efficiency, dependability, stimulation and novelty.

| Item | Scale | Attribute | Rating | | | Attribute |
|------|-------|--------------------|--------|-----|---|----------------------------|
| | | | 1 | ... | 7 | |
| 1 | A | Annoying | o | ... | o | enjoyable |
| 2 | P | not understandable | o | ... | o | understandable |
| 3 | N | Creative | o | ... | o | dull |
| 4 | P | easy to learn | o | ... | o | difficult to learn |
| 5 | S | Valuable | o | ... | o | inferior |
| 6 | S | Boring | o | ... | o | exciting |
| 7 | S | not interesting | o | ... | o | interesting |
| 8 | D | Unpredictable | o | ... | o | predictable |
| 9 | E | Fast | o | ... | o | slow |
| 10 | N | Inventive | o | ... | o | conventional |
| 11 | D | Obstructive | o | ... | o | supportive |
| 12 | A | Good | o | ... | o | bad |
| 13 | P | Complicated | o | ... | o | easy |
| 14 | A | Unlikable | o | ... | o | pleasing |
| 15 | N | Usual | o | ... | o | leading edge |
| 16 | A | Unpleasant | o | ... | o | pleasant |
| 17 | D | Secure | o | ... | o | not secure |
| 18 | S | motivating | o | ... | o | demotivating |
| 19 | D | meets expectations | o | ... | o | does not meet expectations |
| 20 | E | inefficient | o | ... | o | efficient |
| 21 | P | clear | o | ... | o | confusing |
| 22 | E | impractical | o | ... | o | practical |
| 23 | E | organized | o | ... | o | cluttered |
| 24 | A | attractive | o | ... | o | unattractive |
| 25 | A | friendly | o | ... | o | unfriendly |
| 26 | N | conservative | o | ... | o | innovative |

As a result of applying the UEQ version described in [65], we obtained end-users' general impressions by evaluating the *Attractiveness* scale. The pragmatic quality aspects that describe interaction qualities related to the goals the end-users aim to achieve when using *parcanbot* are evaluated by the *Perspicuity*, *Efficiency* and *Dependability* attributes. The hedonic qualities are related to pleasure or fun while using *parcanbot*. These are obtained by evaluating the *Stimulation* and *Novelty* attributes. To compare if *parcanbot* outperforms the traditional interface built on the top of the CKAN portal, the UEQ evaluations of both interface tools are compared based on the averages for each UEQ scale.

The UEQ evaluations were performed with 25 end-users, given that a stable measurement result can be obtained by applying the UEQ to 20-30 end-users [63]. The process for all of them included four steps:

- *Demonstration.* They were shown how to use *parcanbot* to obtain answers on procedurally driven dynamics in plenary sessions on what happened (what legislative initiative had been debated and procedural actions) as well as transparency and accountability information on members and parliament (e.g., news, agenda, commissions, legislative initiatives presented, interventions and salary).
 - *Practice.* A practice task was designed to get familiar using both *parcanbot* and the interface of CKAN. The practice task consisted of obtaining the answers to five different information needs on transparency and accountability. The questions asked were: (1) What are the legislative initiatives presented by a specific member of parliament for two specific legislatures; (2) What is the salary in a legislature for a specific member; (3) What has been the agenda of a specific member for a specific legislature? (4) Is there a citizen participation draft law on Libraries in the Canary Islands? If it exists, what does it consist of? and (5) What were the most relevant parliamentary news items during the last two years?
 - *UEQ Analysis.* After the previous practice task, each end-user had to evaluate *parcanbot* and the traditional interface on CKAN. For each interface tool, the participants filled out the UEQ with 26 questions (see Table 2). The main aim was to check the *parcanbot* end-user experience and if it gives a better user experience compared to the conventional interface on the open data portal CKAN. The data analytical approach was performed with the tool available on the UEQ web site¹³ that automatically calculates the scale values and creates a bar chart to visualize the results for both interfaces. The analytical tool scaled the items from -3 to +3, where -3 represents the most negative answer, 0 a neutral answer, and +3 the most positive answer. In this context, values above +1 indicate a positive impression of the end-users concerning this scale, values below -1 a negative impression [64].
- The two UEQs filled out by participants were analyzed by comparing the quantitative data of the corresponding scales and items for both questionnaires. The chart in the Figure 8 shows the results for the comparison of *parcanbot* and the interface on CKAN concerning the UEQ scales. It shows that *parcanbot* created an overall positive impression concerning all the scales of the UEQ: attractiveness, perspicuity, efficiency, dependability, stimulation and novelty with high values for all of them. *Parcanbot* outperforms CKAN portal in all these scales. Analyzing the item results of the attractiveness scale, the interface of CKAN is considered less attractive compared with *parcanbot*. This is motivated because CKAN portal is perceived as less enjoyable, pleasant, attractive and friendly compared to *parcanbot*.

¹³<http://www.ueq-online.org/>

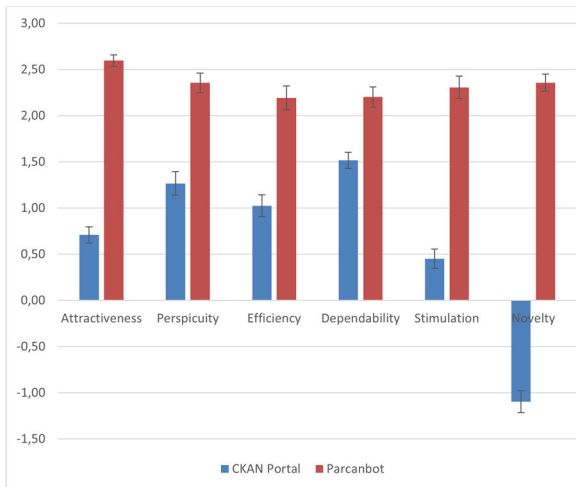


FIGURE 8. Comparison of CKAN portal interface with *Parcanbot* concerning the UEQ scales.

Concerning the perspicuity scale, participants considered that both interfaces are similar to learn in terms of use. However, they considered that *parcanbot* is more understandable and easier to use compared to the CKAN portal. Regarding the efficiency scale, most of the participants responded that both interfaces can be considered as practical tools. However, *parcanbot* was perceived as faster, more efficient and organized in comparison with the CKAN portal. With regard to the dependability scale, participants considered that both interfaces were equally secure, but they believed that *parcanbot* exceeded their expectations, and it was more supportive compared to the CKAN portal. The answers also revealed that for most participants, the CKAN portal was more predictable when using it. This could be because it is based on the traditional method of searching and navigating data in the portal using keywords and does not include natural language and other communication modalities such as speech. The answers corresponding to the items linked to the hedonic scales stimulation and novelty had a negative impression with the CKAN portal because the participants considered that navigating through the portal is less exciting, interesting and motivating compared to *parcanbot* although both interfaces were perceived as valuable tools. The answers also revealed that *parcanbot* was seen as more innovative, inventive, creative and developed with leading-edge technology in relation to CKAN portal.

- **Feedback gathering.** After the participants filled out the UEQ, a short questionnaire with a specific rating question was performed to obtain information about the most valuable data to query on *parcanbot*. The items were rated on a 7-point Likert scale with the endpoints *Not important at all* (1) and *Very important* (7). In addition, observational findings were achieved from free comments concerning both interfaces to

obtain end-users' feedback for future improvement of *parcanbot*.

The results of the short questionnaire rating *parcanbot* on the most valuable data revealed that they are data related to legislative proposals, voting results, decision-making and deputies' salaries. After reviewing the comments concerning end-users' feedback, we have acquired the following insights: (1) when the number of datasets is reduced and the amount of data inside them is also scarce, both approaches are useful, although *parcanbot* is perceived as a new digital channel that helps improve the communication between the parliament and citizens; (2) compared to the existing open data portal CKAN, *parcanbot* is more useful, faster and easier to use to obtain the information needed using natural language when the number of datasets is large; (3) using natural language with speech communication modality as a communication channel is more engaging than navigating through an open data portal to understand legislation. As one participant stated, "it takes several clicks to get to where you need to be when navigating through an open data portal, I prefer speech as a technique to locate data"; (4) when the amount of data in a dataset is large, it would be recommendable to enable visual data in *parcanbot* to help the understanding of data more quickly; (5) to obtain more value from data, it would be useful to combine parliamentary datasets with datasets from other domains to obtain the effects and benefits of the proposed laws over time; (6) *parcanbot* can be considered a valuable tool to reduce the transparency-accountability gap when parliamentary open data is provided by the parliament; and (7) most end-users also highlighted that *parcanbot* helps enable access to lawmaking information in a friendly way, which is perceived not only by easier use of published data but also as a support to get more actively involved in the practice of lawmaking by subsequent collaboration in participatory processes.

C. TESTING

The process of testing is used to assess the correct behavior of the proposed core component of the architecture. That is, the API ecosystem. This process is essential for the governance of all API resources in the ecosystem. It is executed when the Publisher Agent receives a percept *health-checking*, implying that the enabled-API resources need to be validated to detect reliable/faulty resources. As a result of this process, the values of each one of the four quality properties (Ql_1 , Ql_2 , Ql_3 and Ql_4) for each resource are obtained (see Section V-A1) and updated in the current internal state of the agent. Then, subject to these values, the agent decides on what action to carry out (i.e., publishing reliable API resources on ODPs, unpublishing faulty API resources on ODPs, and notifying faulty resources to ecosystem's API producers).

In this context, testing the API ecosystem requires the generation of different test cases with high code coverage

and that can determine the reliability and performance of all available API resources. To define different test cases, we need to consider the software components involved in the Web framework on which the API resources have been developed. Given the API resources have been built on Django REST framework (DRF), three different components need to be considered: managers, serializers and views. Managers are the interfaces through which database query operations are provided to DRF. Serializers are used to transform the results of database queries into the format required to be shown when the API resource is consumed, while the views are the mechanisms that DRF uses to be able to render the given information and display it through a path.

According to these three components, we define five test cases to test separately each component and to ensure that all API resources behave correctly using both correct and incorrect data inputs and a sixth test case to assess performance results:

- *Nominal manager-based test cases*: these test cases assess the accurate behavior of query operations performed by each API resource to retrieve the correct data from the corresponding databases.
- *Nominal serializer based test cases*: here it is tested whether the serializer components of each API resource behaves correctly, transforming the data collected through database queries into the format that is required.
- *Faulty serializer based test cases*: for each API resource, missing information is generated and sent to the serializer to assess that the code error is generated.
- *Nominal and faulty view-based test cases*: this tests if each API resource behaves correctly with the view components. That is, if it sends the required information correctly through the URL designed for the consumption of this API resource. A status code 2xx is returned if everything behaves correctly, otherwise a 4xx family code is returned.
- *Nominal and faulty parameter-based test cases*: for each API resource and each parameter, a nominal test case with the correct data type and required parameter is tested to assess that a status code 2xx is returned. On the other hand, for each API resource and each parameter, a faulty test is performed with wrong data types and missing parameter values to assess that a 4xx or 5xx family code error is returned.

The results of the above five test cases were used to determine the value of the quality property Ql_1 by means of (1), where this value is used in turn to calculate the value of the quality property Ql_2 by means of (2). The values of n in (2) and $\delta_{publishing}$ in (4) were respectively set to 10 and 7. This allows us to determine the reliability of each API resource considering the history of the n previous environment states to the current environment state s_t and taking into account that more than $n - \delta_{publishing}$ testing cases must not be passed (in a consecutive or non-consecutive way) for an

unpublishing action of an API resource by the Publisher Agent in (4).

Currently, on average, the generated test cases obtained 100%, 99% and 73% of coverage respectively for the managers, serializers and views components of RDF as well as a 100% of coverage for test cases assessing the correct behavior when parameter values are used by the API resource.

- *Performance-based test cases*: A sixth test case based on the response rate was conducted to ensure that each API resource is robust and performant. This test case consisted of three different HTTP requests with the GET operation on the same resource. When a test case is executed, it returns the average value of the different HTTP requests. The result obtained for each test case executed is used to calculate the quality property Ql_3 of each API resource, which is utilized by the Publisher Agent to notify faulty resources to ecosystem's API producers by means of (6). The response time threshold, $\gamma_{response}$, was set by performing a preliminary study and analyzing the response rate in terms of the average value of response of each API resource, where the average higher value of all of them was considered to establish the threshold. From this study, it was fixed at 5 seconds.

In summary, after two years of testing the API ecosystem in the parliamentary setting of the Parliament of Canary Islands, we have observed that:

- The ecosystem is scalable and generates, as well as publishes, automatically dynamic data when they are made available by the institution. To date, the deployed solution contains more than 3,000 datasets correctly published by the Publisher Agent on the CKAN platform. There were 2,979 datasets automatically generated by 29 variable APIs by the Publisher Agent, while only 22 datasets were generated by 22 fixed APIs. This has eliminated the tedious and manual process of developing API resources by ecosystem's API producers, which not only eliminates errors but also leads to significant cost savings.
- The ecosystem is stable and robust. No fault has been reported either in the nominal tests or in the faulty tests for the ecosystem in the case of variable API resources. This is because all of them were generated correctly by the Publisher Agent. Only one unpublished action was reported on a fixed API on awards datasets as consequence of a specific software modification in the middleware, which was solved immediately as the notification was received by ecosystem's API producers. This shows that all resources have been correctly published.
- APIs are performant. This has been evaluated by analyzing their response time during the use of the system for over two years. Only two notifications about faulty resources have been reported from the Publisher Agent to ecosystem's API producers.

This was motivated by a communication network problem that supports the API ecosystem infrastructure. Given the problem was resolved immediately, no resource was unpublished by the agent.

- The statement coverage of the generated test cases is on average 93% considering the different software components included in the definitions. This demonstrates the reliability of all available API resources.

VII. CONCLUSION

In this paper, we present SuDaMa, an OGD management framework to address the significant challenges of long-term data publishing and consumption with the aim of: (1) automating the publishing/unpublishing of OGD on open data platforms, which need to be updated dynamically from data systems that operate continuously, (2) providing an easier and more usable way for both developers and end-users to access OGD, and 3) improving the consumption experience by introducing conversational bots. This framework has been created and validated in the context of an open data innovation project with the Parliament of the Canary Islands. As a result, a sustainable OGD management system has been developed and evaluated. The deployed solution remains operational and is being continuously validated by the Parliament with more than 3,000 datasets published by the API ecosystem after two years of commissioning.

The main findings we have obtained from our research results during all this time from the wide diversity of actors involved are:

- From the IT CIO, practitioners and stakeholders' standpoints, the framework and the resulting data management system presented is considered a holistic and remarkable solution that can assist parliaments in their digital transformation in OGD long-term publishing and consumption. It will lead to greater innovative potential of parliamentary service provision and enhanced policymaking. Moreover, the proposed solution is fully aligned with the recommendations of the EU Commission priorities, Member State policies and related standards. Special attention has been paid to the principles recommended by EU directives for the design of API resources using open-source tools by default for such development. The IT CIO also highlighted that in a parliamentary domain, which is continuously changing over time, automation of the API ecosystem by the autonomous agent is essential to ensure the sustainability of an OGD management system. On the one hand, this automation guarantees long-term publishing in terms of providing timely data as well as easy and fast access to them. On the other hand, it eliminates development errors in publishing/unpublishing tasks and lowers development and maintenance costs. Moreover, given APIs are a general purpose, domain-neutral technology, the solution presented can be applied to a huge number of domains that use dynamic data: smart cities, citizen

science, health, mobility, statistics, meteorological data, agriculture, energy, and skills/jobs data.

- From parliamentary practitioners and ULL researchers' viewpoints, internal and external benefits were identified which included innovation triggering, efficiency gains and improving access to and use of parliamentary open data assets, as well as the promotion of digital ecosystems and new economic opportunities. In addition, facilitating access by an API ecosystem has had an impact on parliament openness and transparency, generating trust among citizens via additional mobile applications built on the API ecosystem to create virtuous feedback loops when citizen engage with parliament.
- From the IT project advisory board's perspective, the API ecosystem has proved to be scalable, reliable and robust, generating automatically dynamic data when they are made available by the institution. This eliminated the tedious and manual process of developing and maintaining API resources, which has had positive effects on the performance of the institution by not only eliminating errors but also lowering related development and maintenance costs, as well as increasing the flexibility of focusing on high level objectives instead of operational ones.
- From software developers' points of view, the solution proposed strongly supports the provision of dynamic data with regular updates, meaning data are released continuously and therefore, can be reused immediately after collection by means of suitable APIs. This ensures faster and easier software development, which can help catalyze new entrepreneurial efforts to implement real-time based applications as well as providing innovative data-enabled products for parliaments, governments, researchers, companies, citizens, journalists, students, NGOs, and intermediaries.
- From end-users' standpoint, the conversational bot, *parcanbot*, meets their expectations. It was perceived as a new digital channel that could help improve communication between the parliament and citizens. It was considered more useful, easier to use and more engaging compared to navigating data through the existing open data portal *CKAN* since parliamentary contexts are associated with huge volumes of information. Indeed, it is considered a valuable tool for reducing the transparency-accountability gap.

APPENDIX

A. LIST OF ACRONYMS

| | |
|-------------|------------------------------------|
| API | Application Programming Interface. |
| DRF | Django REST Framework. |
| ETL | Extract-Transform-Load. |
| EU | European Union. |
| HCI | Hyper Converged Infrastructure. |
| JSON | JavaScript Object Notation. |

| | |
|---------------|--|
| OAI | Open API Initiative. |
| ODP | Open Data Platform. |
| OGD | Open Government Data. |
| OGP | Open Government Partnership. |
| PDDL | Public Domain Dedication and Licence. |
| RAML | RESTful API Modelling Language. |
| REST | Representational State Transfer. |
| RO | Reference Ontology. |
| SuDaMa | Sustainable Open Government Data Management Framework for long-term and consumption. |
| TA | Technology Assessment. |
| UEQ | User Experience Questionnaire. |
| UX | User experience. |

REFERENCES

- [1] Open Government Group. (2007). *Open Government Data Principles*. Accessed: Jul. 2021. [Online]. Available: <https://opengovdata.org/>
- [2] J. Attard, F. Orlandi, S. Scerri, and S. Auer, "A systematic review of open government data initiatives," *Government Inf. Quart.*, vol. 32, no. 4, pp. 399–418, Oct. 2015, doi: [10.1016/j.giq.2015.07.006](https://doi.org/10.1016/j.giq.2015.07.006).
- [3] R. Enriquez-Reyes, S. Cadena-Vela, A. Fuster-Guillo, J.-N. Mazon, L. D. Ibanez, and E. Simperl, "Systematic mapping of open data studies: Classification and trends from a technological perspective," *IEEE Access*, vol. 9, pp. 12968–12988, 2021, doi: [10.1109/ACCESS.2021.3052025](https://doi.org/10.1109/ACCESS.2021.3052025).
- [4] Y. Gao, M. Janssen, and C. Zhang, "Understanding the evolution of open government data research: Towards open data sustainability and smartness," *Int. Rev. Administ. Sci.*, vol. 4, pp. 1–17, Apr. 2021, doi: [10.1177/00208523211009955](https://doi.org/10.1177/00208523211009955).
- [5] H. Jiang, Q. Shao, J. J. H. Liou, T. Shao, and X. Shi, "Improving the sustainability of open government data," *Sustainability*, vol. 11, no. 8, p. 2388, Apr. 2019, doi: [10.3390/su11082388](https://doi.org/10.3390/su11082388).
- [6] T. Sasse, A. Smith, E. Broad, J. Tennison, P. Wells, U. Atz, W. Carrara, and H. Bollers, *Recommendations for Open Data Portals: From Setup to Sustainability*. London, U.K.: The Open Data Institute, 2020. [Online]. [Online]. Available: https://data.europa.eu/sites/default/files/edp_s3wp4_sustainability_recommendations.pdf
- [7] C. Batini, C. Cappiello, C. Francalanci, and A. Maurino, "Methodologies for data quality assessment and improvement," *ACM Comput. Surv.*, vol. 41, no. 3, pp. 1–52, Jul. 2009, doi: [10.1145/1541880.1541883](https://doi.org/10.1145/1541880.1541883).
- [8] R. Máchová and M. Lnenická, "Evaluating the quality of open data portals on the national level," *J. Theor. Appl. Electron. Commerce Res.*, vol. 12, no. 1, pp. 21–41, 2017.
- [9] B. Fan and Y. Zhao, "The moderating effect of external pressure on the relationship between internal organizational factors and the quality of open government data," *Government Inf. Quart.*, vol. 34, no. 3, pp. 396–405, Sep. 2017.
- [10] M. Lnánicka, "An in-depth analysis of open data portals as an emerging public e-service," *Int. J. Social, Behav., Educ., Econ. Manage. Eng.*, vol. 9, no. 2, pp. 589–599, Apr. 2015.
- [11] D. S. Sayogo, T. A. Pardo, and M. Cook, "A framework for benchmarking open government data efforts," in *Proc. 47th Hawaii Int. Conf. Syst. Sci. (HICSS)*, Jan. 2014, pp. 1896–1905.
- [12] N. Veljković, S. Bogdanović-Dinić, and L. Stoimenov, "Municipal open data catalogues," in *Proc. Conf. E-Democracy Open Government*, 2011, pp. 195–207.
- [13] N. Veljković, S. Bogdanović-Dinić, and L. Stoimenov, "Benchmarking open government: An open data perspective," *Government Inf. Quart.*, vol. 31, no. 2, pp. 278–290, Apr. 2014.
- [14] European Commission. (Jun. 2019). *Directive (EU) 2019/1024 of the European Parliament and of the Council on open data and the re-use of public sector information*. Accessed Jul. 2021. [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=uriserv:OJ.L_.2019.172.01.0056.01.ENG
- [15] European Commission. (Feb. 2020). *A European strategy for data*. Accessed: Jul. 2021. [Online]. Available: https://ec.europa.eu/info/sites/default/files/communication-european-strategy-data-19feb2020_en.pdf
- [16] European Commission. (Nov. 2018). *Commission Delegated Regulation EU 2019/411*. Accessed: Jul. 2021. [Online]. Available: <https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32019R0411&from=EN>
- [17] O. A. Sennaik, M. Waqar, E. Osagie, I. Hassan, A. Stasiewicz, L. Porwol, and A. Ojo, "Towards intelligent open data platforms: Discovering relatedness in datasets," in *Proc. Intell. Syst. Conf. (IntelliSys)*, Sep. 2017, pp. 414–421, doi: [10.1109/IntelliSys.2017.8324327](https://doi.org/10.1109/IntelliSys.2017.8324327).
- [18] S. Park and J. R. Gil-Garcia, "Open data innovation: Visualizations and process redesign as a way to bridge the transparency-accountability gap," *Government Inf. Quart.*, vol. 25, May 2021, Art. no. 101456, doi: [10.1016/j.giq.2020.101456](https://doi.org/10.1016/j.giq.2020.101456).
- [19] A. Vetrá, L. Canova, M. Torchiano, C. O. Minotas, R. Iemma, and F. Morando, "Open data quality measurement framework: Definition and application to open government data," *Government Inf. Quart.*, vol. 33, no. 2, pp. 325–337, Apr. 2016, doi: [10.1016/j.giq.2016.02.001](https://doi.org/10.1016/j.giq.2016.02.001).
- [20] W. Tan, Y. Fan, A. Ghoneim, M. A. Hossain, and S. Dustdar, "From the service-oriented architecture to the web API economy," *IEEE Internet Comput.*, vol. 20, no. 4, pp. 64–68, Jul. 2016, doi: [10.1109/MIC.2016.74](https://doi.org/10.1109/MIC.2016.74).
- [21] R. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, Dept. Inf. Comput. Sci., Univ. California Irvine, 2000.
- [22] The Linux Foundation. *The Open API Specification Version 3.0.2*. Accessed: Jul. 2021. [Online]. Available: <http://spec.openapis.org/oas/v3.0.2>
- [23] *RAML: RESTful API Modeling Language Specification 1.0*. Accessed: Jul. 2021. [Online]. Available: <https://raml.org/>
- [24] F. Ahmadi Zeleti and A. Ojo, "Qualitative structural model for capabilities in open data organizations," in *Proc. 52nd Hawaii Int. Conf. Syst. Sci.*, Jan. 2019, pp. 2902–2911.
- [25] J. Fawcett, "Examining open data as a source of competitive advantage for big businesses," in *Proc. Open Data Res. Symp.*, 2016, pp. 1–19.
- [26] A. Zuidervijk and M. Janssen, "Participation and quality in open data use: Open data Infrastructures evaluated," in *Proc. 15th Eur. Conf. E-Government*, 2015, pp. 351–359.
- [27] L. Bernzten, M. Johannessen, K. Andersen, and J. Crusoe, "Parliamentary open data in scandinavia," *Computers*, vol. 8, no. 3, p. 65, Sep. 2019, doi: [10.3390/computers8030065](https://doi.org/10.3390/computers8030065).
- [28] V. Lopez, S. Kotoulas, M. Sbodio, M. Stephenson, A. Gkoulas-Divanis, and P. M. Aonghusa, "QuerioCity: A linked data platform for urban information management," in *Proc. 11st Int. Semantic Web Conf.*, vol. 7650, 2012, pp. 148–163.
- [29] F. Gao, M. I. Ali, and A. Mileo, "Semantic discovery and integration of urban data streams," in *Proc. 5th Workshop Semantics Smarter Cities*, 2014, pp. 15–30.
- [30] H. Santos, P. P. Pinheiro, and D. L. McGuinness, "Contextual data collection for smart cities," in *Proc. 6th Workshop Semantics Smart Cities*, 2015, pp. 1–16.
- [31] S. Bischof, A. Polleres, and S. Sperl, "City data pipeline," in *Proc. I-SEMANTICS Posters Demonstrations Track*, 2013, p. 45.
- [32] DuraSpace Community. *VIVO*. Accessed: Sep. 2021. [Online]. Available: <https://duraspace.org/vivo/>
- [33] F. Darari and R. Manurung, "LinkedLab: A linked data platform for research communities," in *Proc. Adv. Comput. Sci. Inf. Syst. (ICACSIS)*, 2011, pp. 253–258.
- [34] F. A. Musyaffa, L. Halilaj, Y. Li, F. Orlandi, and H. Jabeen, "Openbudgets. Eu: A platform for semantically representing and analyzing open fiscal data," in *Proc. Int. Conf. Web Eng.*, vol. 10845, 2018, pp. 433–447.
- [35] X. Liu, A. Heller, and P. S. Nielsen, "CITIESData: A smart city data management framework," *Knowl. Inf. Syst.*, vol. 53, no. 3, pp. 699–722, Dec. 2017, doi: [10.1007/s10115-017-1051-3](https://doi.org/10.1007/s10115-017-1051-3).
- [36] P. Vassiliadis, "A survey of extract-transform-load technology," *Int. J. Data Warehousing Mining*, vol. 5, no. 3, pp. 1–27, 2007.
- [37] X. Masip-Bruin, G.-J. Ren, R. Serral-Gracia, and M. Yannuzzi, "Unlocking the value of open data with a process-based information platform," in *Proc. IEEE 15th Conf. Bus. Informat.*, Jul. 2013, pp. 331–337, doi: [10.1109/CBI.2013.54](https://doi.org/10.1109/CBI.2013.54).
- [38] R. Eckelberg, V. B. Calixto, and M. H. Pimentel, "Educational open government data: From requirements to end users," in *Proc. Int. Conf. Web Eng.*, vol. 10845, 2018, pp. 463–470.
- [39] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. London, U.K.: Pearson, 2010.

- [40] I. Poola, "Making artificial intelligence (AI) and disrupted business intelligence (BI) truly conversational with humanity touch, automated descriptions and talking bots," *Int. J. Adv. Res., Ideas Innov. Technol.*, vol. 3, no. 5, pp. 573–577, 2017.
- [41] L. C. Klopfenstein, S. Delpriori, S. Malatini, and A. Bogliolo, "The rise of bots: A survey of conversational interfaces, patterns, and paradigms," in *Proc. Conf. Designing Interact. Syst.*, Jun. 2017, pp. 555–565.
- [42] A. Androutsopoulou, N. Karacapilidis, E. Loukis, and Y. Charalabidis, "Transforming the communication between citizens and government through AI-guided chatbots," *Government Inf. Quart.*, vol. 36, no. 2, pp. 358–367, Apr. 2019.
- [43] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A design science research methodology for information systems research," *J. Manage. Inf. Syst.*, vol. 24, no. 3, pp. 45–77, 2007, doi: 10.2753/MIS0742-1222240302.
- [44] 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.
- [45] K. Munir and M. Sheraz Anjum, "The use of ontologies for effective knowledge modelling and information retrieval," *Appl. Comput. Informat.*, vol. 14, no. 2, pp. 116–126, Jul. 2018.
- [46] M. I. S. Oliveira and B. F. Lóscio, "What is a data ecosystem?" in *Proc. 19th Annu. Int. Conf. Digit. Government Res., Governance Data Age*, Delft, The Netherlands, May 2018, pp. 1–6.
- [47] C. Brewster and K. O'Hara, "Knowledge representation with ontologies: The present and future," *IEEE Intell. Syst.*, vol. 19, no. 1, pp. 72–81, Jan. 2004.
- [48] A. Immonen, E. Ovaska, and T. Paaso, "Towards certified open data in digital service ecosystems," *Softw. Qual. J.*, vol. 26, no. 4, pp. 1257–1297, 2018.
- [49] T. Davies, "Open Data: Infrastructures and ecosystems," *Open Data Res.*, vol. 18, pp. 1–6, Jan. 2011.
- [50] M. Heimstadt, F. Saunderson, and T. Heath, "Conceptualizing Open Data ecosystems: A timeline analysis of Open Data development in the UK," School Bus. Econ., Free Univ. Berlin, Berlin, Germany, Tech. Rep. 2014/12, 2014.
- [51] D. Lee, "Building an open data ecosystem—An Irish experience," in *Proc. ICEGOV*, Guimaraes, Portugal, pp. 351–360.
- [52] H. Ed-douibi, J. L. Canovas Izquierdo, and J. Cabot, "Automatic generation of test cases for REST APIs: A specification-based approach," in *Proc. IEEE 22nd Int. Enterprise Distrib. Object Comput. Conf. (EDOC)*, Oct. 2018, pp. 181–190.
- [53] Internet Engineering Task Force. (2012). *URI Template specification RFC 6570*. Accessed: Jul. 2021. [Online]. Available: <https://tools.ietf.org/html/rfc6570>
- [54] M. Wooldridge, *An Introduction to MultiAgent Systems*. Hoboken, NJ, USA: Wiley, 2009.
- [55] T. Davis. *What is a WebHook*. Accessed: Jul. 2021. [Online]. Available: <https://webhooks.pbworks.com/w/page/13385124/FrontPage>
- [56] E. Sánchez-Nielsen, F. Chávez-Gutiérrez, and J. Lorenzo-Navarro, "A semantic parliamentary multimedia approach for retrieval of video clips with content understanding," *Multimedia Syst.*, vol. 25, no. 4, pp. 337–354, Aug. 2019.
- [57] (2020). *Protégé*. [Online]. Available: <https://protege.stanford.edu/>
- [58] (2018). *EU General Data Protection Regulation*. Accessed: Jul. 2021. [Online]. Available: https://ec.europa.eu/info/law/law-topic/data-protection/data-protection-eu_en
- [59] L. Sweeney, "Achieving K-anonymity privacy protection using generalization and suppression," *Fuzziness Knowl.-Based Syst.*, vol. 10, no. 5, pp. 571–588, 2002.
- [60] A. Engel, *Verification, Validation and Testing of Engineered Systems*. Hoboken, NJ, USA: Wiley, 2010.
- [61] D. Banta, "What is technology assessment?" *Int. J. Technol. Assessment Health Care*, vol. 25, no. S1, pp. 7–9, Jul. 2009, doi: 10.1017/S0266462309090333.
- [62] M. Schrepp, M. A. Hinderks, and J. Thomaschewski, "Applying the user experience questionnaire (UEQ) in different evaluation scenarios," in *Design, User Experience, and Usability. Theories, Methods, and Tools for Designing the User Experience* (Lecture Notes in Computer Science), vol. 8517, A. Marcus, Ed. Berkeley, CA, USA: Springer, 2014, pp. 383–392.
- [63] M. Schrepp, A. Hinderks, and J. Thomaschewski, "Construction of a benchmark for the user experience questionnaire (UEQ)," *Int. J. Interact. Multimedia Artif. Intell.*, vol. 4, no. 4, pp. 40–44, 2017.
- [64] M. Rauschenberger, M. Schrepp, M. Cota, and J. Thomaschewski, "Efficient measurement of the user experience of interactive products. How to use the user experience questionnaire (UEQ)," *Int. J. Interact. Multimedia Artif. Intell.*, vol. 2, no. 1, pp. 39–45, 2013.
- [65] E. Sánchez-Nielsen and F. Chávez-Gutiérrez, "Using semantic annotations on political debate videos for building open government based lawmaking," *Expert Syst.*, vol. 21, Jun. 2021, Art. no. e12748, doi: 10.1111/exsy.12748.



ELENA SÁNCHEZ-NIELSEN received the B.Sc. degree in computer science from the Universidad de Las Palmas de Gran Canaria, Spain, in 1994, and the M.Sc. and Ph.D. degrees in computer science and artificial intelligence from the Universidad de La Laguna, Spain, in 1999 and 2003, respectively. She is currently an Associate Professor with the Departamento de Ingeniería Informática y de Sistemas, Universidad de La Laguna. She has directed several research projects and is often contracted for innovation projects in industry and government. She has published over 80 articles, including in peer-reviewed international journals, book chapters, and conference proceedings. She also serves the community on various technical committees. Her current research interests include the intersection between artificial intelligence and data systems, smart information systems, innovation in eGovernment, which draws on artificial intelligence, and big, open, and linked data systems.



ALEJANDRO MORALES received the B.Sc. degree from the Universidad de La Laguna, Spain, in 2018, where he is currently pursuing the M.Sc. degree. He is also an Information and Technology Scholar with the Canary Islands Parliament, Spain, where he is working on an open data initiative for the Canary Islands Parliament. His current research interests include application programming interfaces, open data systems, machine learning, and software engineering.



OMAR MENDO received the B.Sc. degree from the Universidad de La Laguna, Spain, in 2019, where he is currently pursuing the M.Sc. degree in machine learning. He is also an Information and Technology Scholar with the Canary Islands Parliament, Spain, where he is working on an open data initiative for the Parliament. His current research interests include machine learning, and open and big data systems.



FRANCISCO CHÁVEZ-GUTIÉRREZ received the B.Sc. degree in computer science from the Universidad de Las Palmas de Gran Canaria, Spain, in 1994, and the M.Sc. and Ph.D. degrees in computer science and artificial intelligence from the Universidad de La Laguna (ULL), in 2007 and 2017, respectively. He is currently the CIO of the Canary Islands Parliament, Spain, and has led all information and technology-related projects at this institution over the last 25 years. He has collaborated with the ULL in research and development projects to innovate information systems in parliamentary settings. He has published various articles, including in peer-reviewed international journals, book chapters, and conference proceedings. His current research interests include eGovernment, innovation in parliamentary information systems, big, open, and linked data systems, and emerging technologies.