Future Generation Computer Systems 75 (2017) 14–29

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Analysis and assessment of a knowledge based smart city architecture providing service APIs

C. Badii, P. Bellini, D. Cenni, A. Difino, P. Nesi *, M. Paolucci

University of Florence, Department of Information Engineering, DISIT Lab[1], Italy

HIGHLIGHTS

- Knowledge base smart city solution.
- Smart City API and their advanced features.
- Assessing Smart City infrastructures.

ARTICLE INFO

Article history:
Received 10 February 2017
Received in revised form
15 April 2017
Accepted 2 May 2017
Available online 6 May 2017

Keywords:
Smart city
Smart city ontology
Smart city API
Smart mobility
Multi-domain smart city
Smart services

ABSTRACT

The main technical issues regarding smart city solutions are related to data gathering, aggregation, reasoning, data analytics, access, and service delivering via Smart City APIs (Application Program Interfaces). Different kinds of Smart City APIs enable smart city services and applications, while their effectiveness depends on the architectural solutions to pass from data to services for city users and operators, exploiting data analytics, and presenting services via APIs.

Therefore, there is a strong activity on defining smart city architectures to cope with this complexity, putting in place a significant range of different kinds of services and processes. In this paper, the work performed in the context of Sii-Mobility smart city project on defining a smart city architecture addressing a wide range of processes and data is presented. To this end, comparisons of the state of the art solutions of smart city architectures for data aggregation and for Smart City API are presented by putting in evidence the usage semantic ontologies and knowledge base in the data aggregation in the production of smart services. The solution proposed aggregate and re-conciliate data (open and private, static and real time) by using reasoning/smart algorithms for enabling sophisticated service delivering via Smart City API. The work presented has been developed in the context of the Sii-Mobility national smart city project on mobility and transport integrated with smart city services with the aim of reaching a more sustainable mobility and transport systems. Sii-Mobility is grounded on Km4City ontology and tools for smart city data aggregation, analytics support and service production exploiting smart city API. To this end, Sii-Mobility/Km4City APIs have been compared to the state of the art solutions. Moreover, the proposed architecture has been assessed in terms of performance, computational and network costs in terms of measures that can be easily performed on private cloud on premise. The computational costs and workloads of the data ingestion and data analytics processes have been assessed to identify suitable measures to estimate needed resources. Finally, the API consumption related data in the recent period are presented.

1. Introduction

The major companies are proposing solutions to make city smarter, focusing on specific set of domains, such as IBM [1,2] on services for citizens, business, transport, communication, water and energy; [3] on governmental, educational, e-health, safety, energy, transport and utilities; CISCO on people, things and data [4], etc. Most of these solutions present a multi-tier architecture ranging from 3 to 6 layers [5]. On the other hand, the

* Corresponding author.
E-mail addresses: claudio.badii@unifi.it (C. Badii), pierfrancesco.bellini@unifi.it (P. Bellini), daniele.cenni@unifi.it (D. Cenni), Angelo.difino@unifi.it (A. Difino), paolo.nesi@unifi.it (P. Nesi), michela.paolucci@unifi.it (M. Paolucci).

[1] http://www.disit.org, http://www.sii-mobility.org.

http://dx.doi.org/10.1016/j.future.2017.05.001

number of tiers is partially relevant to the transformation of data in value for business, and thus to services for the city users, and in opportunities for the enterprises and city operators interested in creating innovative and effective services, while exploiting city data and information [6–8]. Also, the smart city ranking models are not suitable in putting in evidence these aspects, since they are mainly focused on counting the number of provided open datasets, smart services, solutions, or energy results [9,10].

As a general consideration, the main technical issues regarding smart city solutions are related to data access, aggregation, reasoning, access and delivering services via Smart City APIs. The final aim is serving city users in a smarter and more efficient manner, stimulating their participation to the city strategies. Therefore, collected and produced data are used to facilitate the creation of smart and effective services exploiting city data and information. Specific end-users' smart services should be developed and managed by enterprises and city operators, rather than by the municipality. On the other hand, the municipality has to provide a flexible data access and services. This means to make effective and efficient the data access with their semantics, the service delivering, the access to define and control dashboards, and the interoperability with any other smart control systems active in the city (e.g., mobility, energy, telecommunication, fire brigade, security, etc.). In the world, municipalities/cities and public administrations are publishing huge amount of open data. These data can be coarsely aggregated for integration by using solutions such as CKAN [11], OpenDataSoft [12], ArcGIS and OpenData [13]. In most cases, these solutions for open data are suitable for collecting open data files and their indexing on the basis of corresponding descriptive metadata. Open data, in those cases, can be uploaded by providing files in different formats: CSV, XLS, XML, SHP, etc. In some cases, they provide access to effective datasets, by using some data integration and visualization tools which provide the possibility of creating graphic charts, such as distributions or pies, on the basis of the values contained in the dataset. In the extreme case, they also provide access to datasets as Linked Data (LD), Linked Open Data (LOD), coding data information in terms of RDF triples [14,15]. Very rarely, they can provide data from some RDF store endpoints to make SPARQL queries on the data exploiting some ontology and other entities [16], rather than working only on metadata. The access to RDF stores for data browsing can be performed by using visual browsers as in [17]. In the case of directly accessible LOD, we are in presence of the so called 5 stars' open data [14]. On the other hand, in most cases the integrated LOD are not supported by multi-domain ontologies, as explained in the sequel. We could state that 6 stars data would also provide a data access and SPARQL queries exploiting a semantic ontology for the integrated data model and data inference [18].

In most cases, the effectiveness of data service system for Smart City is enabled by the availability of private data owned and managed by *City Operators* addressing specific domains: mobility operator, energy providers, business services (health, water), telecom operators, tourist operators, universities, etc. They are the city stakeholders providing data and services with different granularities and sizes. For example, in the city, we can have few energy operators with capillary house distribution, many public transport operators with thousands of vehicles/buses, some telecom operators deploying in the city from tens to hundred thousand or millions of sensors. Different granularity implies different methods for collecting and for providing access to data such as publication of open data files and/or statistics, publication of real time data with consumptions, mobility flow, energy, weather, etc.

Real-time data are provided by city operators through some APIs as Web Services or REST calls. The APIs for providing data to the data aggregator of the city may be compliant with multiple standards (such as DATEX II for mobility, intelligent transport system [19]) for public services, parking; IETF [20], ETSI [21] or OneM2M [22] for Internet of Things (IOT), Green Button Connect [23] for energy data collection. On the other hand, some of the peripheral data kinds collected are not supported by any standard, thus custom solutions are adopted, such as the status of hospitals' emergency units (triage), the status of earthquakes in the regional area, etc. In the E015 digital ecosystem, in order to cope with a large number of APIs in the city, an effort was made, for setting up an environment for collecting documentation about available data via services/APIs and interfaces in the city, by using a web portal [24]. This kind of approach is not solving the lack of data and service/API interoperability since every mobile or Web App developer needs to integrate multiple datasets has to access them by using multiple protocols with the providers. In many cases, also multiple agreements since each offered API may have its own licensing model. Thus, the developers may collect data that still need to be aggregated to make them semantically uniform, referring to the same elements in the city, and to establish multiple agreements.

The effective deploy of smart services for city users is very frequently viable only by exploiting the semantic integration of data as: open data, private data and real time data coming from administrations and different city operators. This implies specific processes of reconciliation and the adoption of unifying data models and ontologies as in Km4City [25]. The semantic aggregation of data coming from several domains is unfeasible without a common ontology, since data are produced by different institutions/companies, by using different formats and aims, different references to geographical elements, and different standards for naming and identification adopted in different moments [26]. Thus, datasets are rarely semantically interoperable each other since have been produced in different time, by different systems, by different people, etc. In addition, they may present different licensing models: some of them can be open, while other may be private of some city operator that would not be interested to lose the ownership by releasing them into an unregulated environment, or could simply provide some restrictions (e.g., no commercial); see for example the data of car sharing companies that are typically private of the company. For open data, as well as for private data, several different licensing models can be adopted [27,28] enabling or preventing some business models, or simply their usage [29].

Therefore, well aggregated and re-conciliated data for the identification of services and locations (open and private) can be exploited by reasoning algorithms for enabling sophisticated service delivering. For example, by **providing** suggestions and hints on rout planning, inter-modality routing, parking, hospital finding in the case of emergence, finding specific point of interests, setting predictions (for parking and traffic) and detecting anomalies for early warning. For example, in [30] a hybrid algorithm combining Fuzzy Logic Controller and Genetic Algorithms has been adopted for reducing the traffic jam.

The data values (actual, predicted and/or detected) can be delivered to different operators and city users by some personal assistants on the basis of the user profile and role. For example, in order to provide information about what is or what would be around a current GPS position, the integration of geographic information and services is needed; while the integration of geolocalized services and the assessment of typical people flows may help the city in improving public services and transport, providing suggestions to the city users, and planning changes in the city [31]. For example, in [32] a solution for estimating the crowd density has been proposed exploiting mobile phones and Bluetooth; while in [33] a solution for monitoring people and vehicles in the city by exploiting multiple data sources has been

proposed. In [34,35], solutions based on Bluetooth server have been proposed for estimating the travel time. Thus, aggregated data can be exploited to implement a large number of services and applications by structuring the Smart City Architecture and the corresponding Smart City APIs.

This paper presents the work performed on defining a smart city architecture and assessing its performance, as developed in the context of Sii-Mobility smart city project. Sii-Mobility aims to provide innovative services for mobility operators and city users moving in the city and in the region, to provide solutions for sustainable mobility and transport systems. In the literature, there are several proposals for smart city architectures, but only few of them are really in place with a relevant range of different kinds of processes, such as addressed in this paper. In particular, specific smart algorithms for data aggregation, personal assistance, solutions for dynamically shaping restricted traffic zones, the production of personalized suggestions for moving city users, aiming at improving sustainable mobility, etc. These requirements necessitated a deep analysis of the state of the art solution, to identify and then develop a solution that allows performing reasoning and deduction on city data collected from city operators, as open data and private data, as static and real time data, as multiple domain data for producing suggestions and stimulus to city users [36]. In this context, the main goals of an innovative and suitable architecture have been: (i) the data aggregation layer to integrate different kinds of data creating a real knowledge base for the city (a sort of expert system with inference capability), (ii) the computational capabilities and process management in the back-end and in the front-end, (iii) the formalization of the Smart City API by which all the web and mobile Apps, and dashboards may have access to the smart city knowledge and services. For point (i) and (iii), a deep comparison among state of the art solutions has been provided. For point (ii), and integrated performance analysis has been performed by using measures that can be extracted from cloud. Thus, the paper is presenting an analysis of the workload of the several smart city services putting in evidence the computational and network workloads of the several areas.

The work presented in this paper has been performed in the context of the Sii-Mobility smart city project on mobility and transport aspects and integrated with city services in general (http://www.sii-mobility.org). Sii-Mobility project includes 24 industrial partners from industry and research, and it has been partially founded by the Italian Ministry of Research as a special national program on Smart City, MIUR SCN. Sii-Mobility is focused on providing innovative services in Tuscany, which is an area of 3.5 million of inhabitants, and it involves the Tuscany Region and several municipalities and local governs in the area for the experimentations, covering almost all the Tuscany region.

The paper is organized as follows. Section 2 presents the comparison of the state of the art of data aggregation solutions. Section 3 presents the Sii-Mobility architecture, scenarios and available data families. The architecture of Sii-Mobility has been defined with the aim of exploiting a knowledge base for reconciliating and aggregating data, and for enabling the spatial, temporal and conceptual reasoning on them. Section 4 presents the assessment performed on the Sii-Mobility smart city architecture in terms of resources consumption over time (computational and network on cloud), analyzing the data ingestion processes, the data analytics processes and the costs for the front end services. Section 5 provides a comparison among state of the art of Smart City APIs vs. proposed Smart City API of Sii-Mobility/Km4City. In addition, a number of examples of the proposed Smart City API are provided. Section 6 reports the assessment about the usage of Sii-Mobility Smart City APIs, identifying the most requested and their impact on the back-end. Conclusions are drawn in Section 7.

2. Smart city architectures analysis and comparison

In general, all the smart city solutions must cope with big data volume, variety, and veracity [37,38]. Open data as static data are not the main source of information in the city. Most of the big data problems connected to the smart city platform are related to real time data as the vehicle and human mobility in the city, energy consumption, health care, and IOT. The smart city architecture should be capable to take advantage of huge amount of big data coming from several domains, at different velocity for exploiting and analyzing them for computing integrated and multidomain information, making predictions, detecting anomalies for early warning and for producing suggestions and recommendations to city users and operators.

In the last years, many architectural solutions have been proposed with the aim of making data accessible, aggregated, usable, and exploitable, etc., and many of them failed in posing the basis for creating a smart city open environment for new and smart applications. It is obvious to state that, cloud and distributed systems approaches are at the basis of the big data solutions provided for smart city, as well as IOT solutions at the basis for collecting data from sensors and devices in the city. On the other hand, the city infrastructure is much more complex, and the limited focus on only some of the above-mentioned aspects would create limitations not accepted by the city operators and limiting the development of city smart services.

Most of the Smart City Architectures are enabling Smart City Services and Applications by setting up and providing different kinds of Smart City APIs. The Smart City Architectures mainly differ each other from the strategy to transform data to services for the city (from data to business), as represented in Figs. 1–3, in which three main architectural approaches are presented. In effect, several hybrid architectural solutions have been proposed in the literature, those depicted in Figs. 1–3 only represent the most important architectural patterns, which will be used in the context of the following discussion. It is worth noting that, when in the rest of the paper we talk about Web and Mobile Apps, we mean applications that exploit data and service accesses, putting them in the hands of City Users in general which can be citizens, tourists, students, commuters, etc. as well as Decision Makers. The applications in the hands of citizens, tourists, students, commuters, may be used to inform, engage, stimulate, and push them to change and/or assume a virtuous behavior according to the city strategies. Moreover, in the following architectural patterns, Cases (a), (b) and (c), the Decision Makers are key personnel of public administrations or city operators interested in extracting from the aggregated and integrated data eventual trends, deductions, data and/or alerts, anomalies, early warnings, about the city. Decision Makers in the city are typically requesting some business intelligent tool for synthetizing results and trends from the huge amount of data as support for making decision. In most cases, they request dashboards for monitoring city trends also considering multidomain aspects as: social media and mobility, weather and traffic, parking conditions and events, etc.

The following cases (from (a) to (c), depicted in Figs. 1–3) can be regarded as incrementally improved smart city architectures.

2.1. Case (a) the info integrator

The Info Integrator of Fig. 1 collects information about APIs provided by different data and/or service providers (including their authentication and licensing), and provides a common place for developers and city operators to browse and learn how to access at the exposed API services and data. Data/service providers can be city operators such as: mobility operators, energy operators, waste and water operators, car sharer, internet of things sensors,

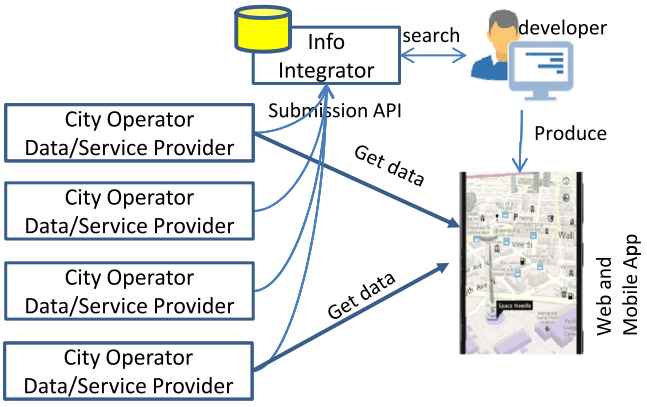


Fig. 1. From data to services of smart city architectures, Case (a) info integrator. The thicker lines carry data streams and data services.

etc. They may provide some open and/or private data, static and/or real time data. In Case (a), the data/services are not integrated each other, each API set allows to access the specific data/service of each single operator. Thus, the API and the data are not semantically interoperable, and the problems in managing the semantic integration of data and services are left in the hands of the developers, that have to cope with different, not harmonized APIs and providers, different authentications, and so on. The developers of applications have to select the data, set up solutions to get them and integrate them every time they change. This also implies that they may have to arrange different contracts and agreements with each data and service provider.

The solutions belonging to this category do not satisfy most of the sub-goals of the EIP document about the Urban Platforms [39] of the European Innovation Partnership, supported by the European Commission, and in general should not be considered Smart City Architectures [40].

On this line, there are many other portals that allow you to publish and promote API, and thus that play the role of Info Integrator as Case (a) if the API provide data about smart city. Examples of the API portals are: https://publications.api.org/, http://www.programmableweb.com/, http://www.apiforthat.com/, https://any-api.com/. Most of them request for a formal documentation for the submission also using a Swagger format (http://swagger.io/). Swagger provides a formal model for presenting API and some tools for assisting developers in their formalization and deploy. On this line, a number of specific APIs on smart city services may be identified. Moreover, the E015 solution for Milan [24], can be regarded as an evolved and specialized example of Case (a) for smart city, where structured information is collected by the City Operators Data/Service provider which must expose their own API documentation via formal E015 Excel files, and it is in turn published on the E015 Web portal. This approach aims to share the knowledge about the API documentation, and thus it may help to regularize the APIs of the city operators, and thus only indirectly facilitates the production of integrated services for city users. The approach of Case (a) do not full activate the passage from "*data to services*" since the data and APIs of the several City Operator are not interoperable, and the licensing agreements consists in multiple relationships arranged between each developer and the several data providers which expose data with their multiple protocols/API. Therefore, the possibility of directly exploiting the data with data visualization tools and/or business intelligence tools, or dashboards for monitoring the city status and the interdependencies among the data coming from different area is very far to be reached. It may be possible to set up some dashboard collecting data from multiple City Operators Data/Service providers, thus partially playing the role of the public administrators.

2.2. Case (b) the data and metadata aggregator

The Data and Metadata Aggregator of Fig. 2 collects data and metadata information (mainly open data) to index and aggregate them in a common model, according to the structure of the open data files as tables in input. It is mainly focused on collecting data in pull calling City Operators' Data/Service Provider (e.g., via REST calls, WS). The resulting aggregated data are made accessible to web and mobile Apps via some automatically generated APIs, disregarding their semantics and the possible matches among entities collected from different datasets that may represent the same elements/entities in the city.

Problems of missed reconciliations are present when datasets provided present multiples locations, dates and time, non-precise identification of city elements, and/or GPS coordinates of different kinds. The data are not re-conciliated each other, and maintain the same quality of the original, so that missing data are still missing. The data integration is mainly performed on the basis of syntactic and lexical forms of data names and content values. The resulting database is typically a set of tables with traditional SQL capabilities. To solve this kind of problem typical data warehouse solutions can be adopted implementing ETL processes for establishing data relationships and improving quality. In some cases, graphic rendering is provided via some data visualization tool, presenting the similar problems of semantic miss match. Thus, the provided APIs are not supported by a domain-oriented ontology. In some cases, a semantic model for data access is provided as marketing strategy for the 5 stars; and thus, they provide SPARQL based API and service. The generated ontological model is a mere representation of tables and does not provide significant inferential support. This also means to have limited reasoning on time and space. Belong to this category of solutions CKAN [11], ArcGIS OpenData [13], OpenDataSoft [12] based on ArcGIS, SOCRATA [41] also based on ArcGIS. The solutions based on ArcGIS provide more capabilities on geospatial queries. Case (b) approaches can be regarded as first level solutions for data interoperability, and can be viable when mainly open data are integrated, without real-time and/or private data. Real Time data may be typically made accessible for pull call or may be provided as stream or push calls as for many IOT solutions. Therefore, when open data have to be integrated with private data and real time data, as in the cases of urban mobility services, the solution has to cope with a number of special cases that have to be separately and manually treated to be integrated with the rest of data; thus, destroying the advantages of integration. Data arriving in stream and push could be collected by separate data brokers, which, in turn, can be called in pull by the data aggregator.

The solutions compliant to Case (b) do not cover all sub-goals of the Urban Platforms [39] since data are not fully harmonized. In [42] the solution of Case (b) has been adopted for realizing Santander smart city, by managing more than 15.000 sensors collected in 1200 sensor nodes. The Santander solution used a number of Data Brokers for harvesting data (playing the role of Data/Service providers), thus data are collected by a big data aggregator. Moreover, the Santander solution presents a CityModel Server which is capable to provide services via City Model API to which web and model applications may connect. CityModel API allows application to perform simple queries, complex queries (making statistical analysis), and subscription (for direct access to the streams in push). According to the analysis presented by [42], the subscription method allows receiving data changes in push, on the other hand in most cases the data are only periodically accessible in pull, and relevant limitations were detected to the lack of a semantic support for reasoning on data. A view of Santander solution for IOT can be recovered on [43], while we should remind that the focus of this paper is on the general

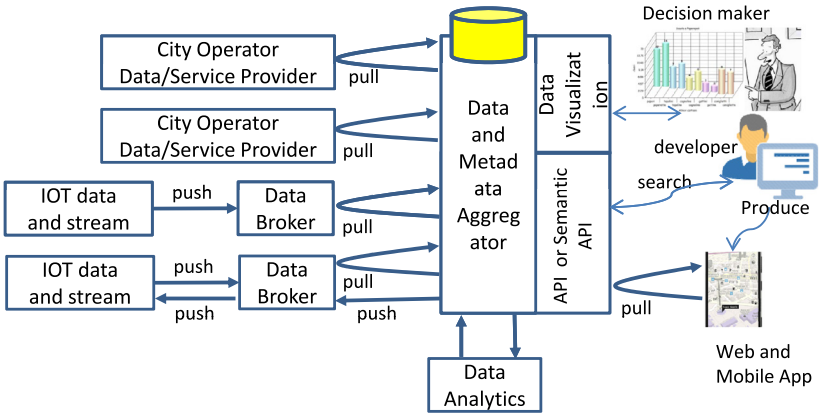


Fig. 2. From data to services of smart city architectures, Case (b) data and metadata aggregator. The thicker lines carry data streams and data services.

architecture and Smart City API. Another example of Case (b) solution, which is mainly focused on data driven aspects collecting data from IOT, has been proposed by SOFIA [6]. Also in this case, some of the exploitation scenarios based on data combinations are hardly addressed since the semantic model has been limited to events and not on composing static and dynamic data streams. The approaches in which the data streams are addressed creating specific data flows may present strong limitations in exploiting the multidomain data relationships which are present in the city [44, 6,45]. Moreover, the IOT solutions are also requesting input from the Smart City system (e.g., to regulate the street light), thus providing the needs of receiving in push data values from the Data Brokers (see Fig. 2). Therefore, when multiple and high volume data flows are processed (analytically and/or semantically) the Data Aggregator should provide services for creating specific processes to satisfy the API requests. Non-trivial cases should consider multiple domain data, contextual static data, and real-time data to produce decision support results in outputs, thus making mandatory the usage of semantically integrated model. Moreover, front-end APIs are substantially in pull since mobile and web Apps call the API, since it is not feasible to the push information on clients for the lack of knowledge about their IP or mobile number. When high volume of requests is received, a caching mechanism has to be set up on the front-end, thus making mandatory the decoupling of data consumption via API vs. data processing/reasoning.

A solution to these problems is the adoption of a Case (c) approach as described in the following.

2.3. Case (c) the semantic aggregator and reasoner

The Semantic Aggregator and Reasoner (see Fig. 3) should collect data and services from the City Operators, to aggregate and integrate them in a unified and semantically interoperable model based on a multi-domain ontology. A basic exploitation of the ontology in the data aggregation may consist in classifying the IOT sensors/actuators which are connected to the IOT Broker via a discovery protocol such as in IOT FIWARE solution [46]. The ontology can be used by the semantic aggregator to model city domains entities and their relationships and not only metadata of data sets and tables as in Case (b). An enhanced exploitation consists in the usage of the ontology as a basis for creating a knowledge base to re-conciliate data to create a coherent model thus reducing the errors, integrating data representing the same concept and coming from different structures, operators, and sources.

The usage of a multi-domain ontology allows the adoption of a model representing relationships of specialization among classes and relationships, aggregation, association, and similarity, that enable the inferential processes in the RDF Graph Database [47, 26,25]. Thus, the obtained knowledge base can be used for creating strategies for data quality improvement and for setting up algorithms and reasoning about the several aspects and services belonging to multiple domains (prediction, early warning, etc.), and geospatial, temporal reasoning. This advantage is also evident on the provided API and in the possibility of providing integrated data and views for the Decision Makers. For the same reason, the obtained Knowledge Base, by populating the ontology with data and inference, can be profitably and easily used for producing smart services such as contextual routing, multimodal contextual routing, suggestions on demand, personal assistants, connected drive, etc.

As described in the following sections, some solutions fit to this Case (c): CitySDK [48] partially covering all features has been developed in an EC project involving major cities and providing specific REST API and grounded on OASC (Open & Agile Smart Cities) adopted the FIWARE NGSI API agnostic model [49]; and more widely covering features Km4City [25] exploited by Sii-Mobility Smart City project, RESOLUTE H2020 Project, and REPLICATE H2020 Project of the European Commission, providing Smarty City API of Km4City as [50,25], and SPUD proposed by IBM in [47] exploiting commercial non open solution via a non-accessible ontology.

Solutions of Case (c) mainly differ from those of Case (b) for the presence of a comprehensive human made ontological model among city entities and not a simple ontology generated on the basis of data tables structure. Moreover, the solutions of Case (c) need mapping tools from data to ontology and to support the reconciliation as performed by DataLift in [51] and by Km4City in [25]. In both cases, vocabularies, algorithms, and dedicated languages have been used, as SILK [52]. It should be noted that the semantic aggregator of Case (c) could be implemented on top of solutions structured as Case (b). In fact, Case (b) solutions may provide facilities listing data sets, updating their acquisition, and accessing to them with some tabular based API automatically generated, and they can be used to feed the semantic model. The solutions of Case (c) are better ranked with respect to the sub-goals of the Urban Platforms [EIP Requirements] covering aspects connected to the harmonization of data, and production of intelligent services. Moreover, the implementation of user experience for value added services (subgoal 5 of EIP) is only accessible in a few of them as analyzed in the following. The above Case (c) solutions have to cope with Graph Database collecting huge amount of data, thus resulting in Big Data cases and scenarios presenting relevant data such as variety, velocity, veracity, volume, etc. [37,47]. An effective integration at semantic level of the data domain enables the creation of Smart Decision Support Systems

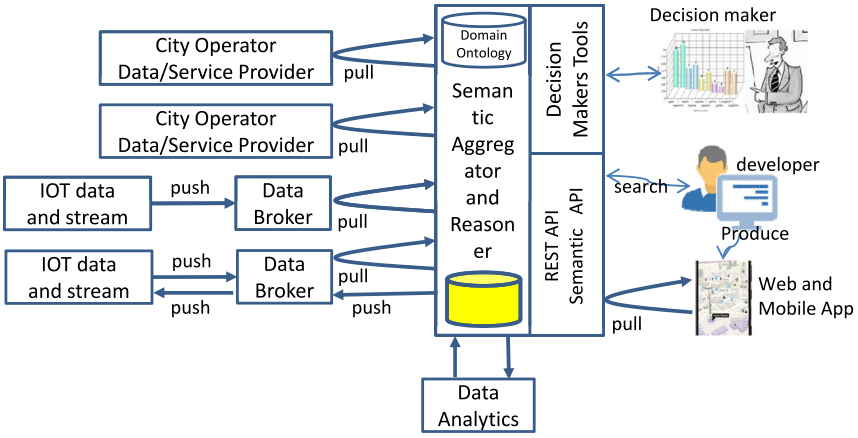


Fig. 3. From data to services of smart city architectures, Case (c) the semantic aggregator and reasoner. The thicker lines carry data streams and data services.

that exploit the possibility of making semantic queries on multiple domains, to make probabilistic reasoning on Bayesian decision support [53,47], and to enable the production of algorithms for implementing personalized routing and Personal Assistants in the city. In some cases, the adoption of graph database to store and retrieve smart city data may be not the most effective solution in terms of performance despite it may enable reasoning as inference [54]. This issue may lead to decide of managing data in multiple data models according to their usage and natural matching between data structure and concepts. For example, semantic relationships among city entities on graph data stores, IOT and time varying data as sensors occurrences in efficient noSQL tabular stores such as CouchDB or MongoDB.

2.4. Architecture comparison

Table 1 summarizes the above presented comparative analysis of the different architectures and Smart City API solutions, to facilitate the city to transform data into services, thus from data to applications for the City Users and for the Decision Makers. As a final consideration, the main differences that can be perceived by the City Users and by the Decision Makers is the number of smart and cross domain services that Case (c) can provide with respect to the other solutions. The Case (c) category of Smart City approaches represents the new generation of solutions, and it is the most interesting area for the study reported in this paper about the resulting APIs, which are more powerful with respect to those provided by the other cases. Case (c) solutions have the potential to cover all aspects, while the effective coverage depends on the coverage provided by the tools adopted and implemented. Among the most relevant tools, we see as the most critical: Data Analytics, Smart City API, and the Decision Maker tools. Each of the them enable different features into the smart city.

3. Sii-mobility scenarios and solution

This section presents the Sii-Mobility (http://www.sii-mobility.org) scenarios and available data, and thus the adopted architecture which is presently in place in the Tuscany area. Sii-Mobility started from the observation that terrestrial transportation systems are more often affected by congestion situations since specific situations cannot be easily foreseen by using traditional Intelligent Transport Systems (ITS), that only cope with transport data and mobility aspects, disregarding events, energy, weather, people flow, etc. The local transportation system presented very high social costs related to the citizens' uneasiness with respect to the available mobility solutions, which are not sustainable and poorly efficient. This is mainly due to the scarce interoperability and intelligence among management and monitoring transport systems, services for mobility, services and systems for goods transportation, ordinances and public services (such as hospital, centers, museums), events, private transport, rail transport, car parks, and moving people, because of the limited capacity of the system to incorporate and react to changes in the city and citizens.

With the aim of producing a smart city infrastructure for solving these problems, Sii-Mobility worked on enabling technologies for smart city and mobility, to integrate and produce the Support of Integrated Interoperability (SII). SII aims to create a smart city big data framework for providing integrated data, supporting services as mobile App, decision support, integrated ticketing, personal assistants, participative portals, crowd sourcing, dashboarding, etc., via Smart City API. It also aims to support Data Analytics and Data Intelligence based on integrated data collected from public administrations open data, private data from operators, and personal data coming from social media and city users. The main objectives of Sii-Mobility are to: (i) reduce social costs of mobility; (ii) simplify the use of mobility systems; (iii) provide solutions for assisting connected drivers/people exploiting inter-modality (connect drive, smart drive or walk); (iv) define solutions for the interoperability with other smart cities' management systems; (v) define solutions for city user engagement and participation and awareness, personalized management of access policies; (vi) study and propose solutions for dynamic management of RTZ (restricted traffic zone) boundaries; (vii) real-time monitoring of supply and demand of public transport. In this paper, the architecture enabling the construction of the Smart City solutions which support the above-mentioned scenarios via Smart City API is discussed and presented. The architecture and solution is presently operative in the Tuscany region, providing services to city users, city operators and public administrators.

As described in the Case (c), architectural solutions are the most suitable for providing smart services as those of Sii-Mobility, provided by mobility operators as routing, personal moving assistant, geolocalized suggestions, personal assistance, decision making support, etc. Therefore, Sii-Mobility.org developed a Smart City architecture as depicted in Fig. 4, which is an evolved Case (c) architecture with semantic aggregation of data and services. The Sii-Mobility architecture collects data from:

- **City Operators and Data Brokers** in pull by using ETL processes which are scheduled on the Big Data processing back office based on DISCES (Distributed Smart City Engine Scheduler) tool developed for Sii-Mobility and made open source. Among the data collected those provided in Open Data from the municipalities, Tuscany region (Observatory of mobility),

Table 1
Comparison of different smart city solutions to pass from data to services. (Y) means yes with limited capabilities, due to the limited data interoperability among the city entities in: time, space, multiple domains (semantic interoperable), structures, services, and relationships.

	Case (a) info integrator	Case (b) data and metadata aggregator	Case (c) semantic aggregator and reasoner
Addressing Open Data	Y	Y	Y
Addressing Private Data	Y	Y	Y
Addressing Real Time Data	Y	Y	Y
Addressing Interoperable Services	(Y)	Y	Y
Collecting data in Push	N	Y	Y
Collecting data in Pull	N	Y	Y
Providing Data Search	N	Y	Y
Providing Metadata Search	Y	Y	Y
Providing Space Reasoning	N	(Y)	Y
Providing Time Reasoning	N	(Y)	Y
Providing Integrated Authenticated Access to data	Only metadata	Y	Y
Providing Syntactic Interoperable Data/Services	N	Y	Y
Providing Semantic Interoperable Data/Services	N	N	Y
API Independent from the Data model changes	N	N	Y
Providing REST API on data	(Y) not integrated	Y	Y
Providing SPARQL API on data	N	(Y)	Y
Providing inference support on Data	N	N	Y
Providing Data Visualization (business intelligence, dashboarding)	N	(Y)	Y
Providing Decision Maker Support	N	(Y)	Y

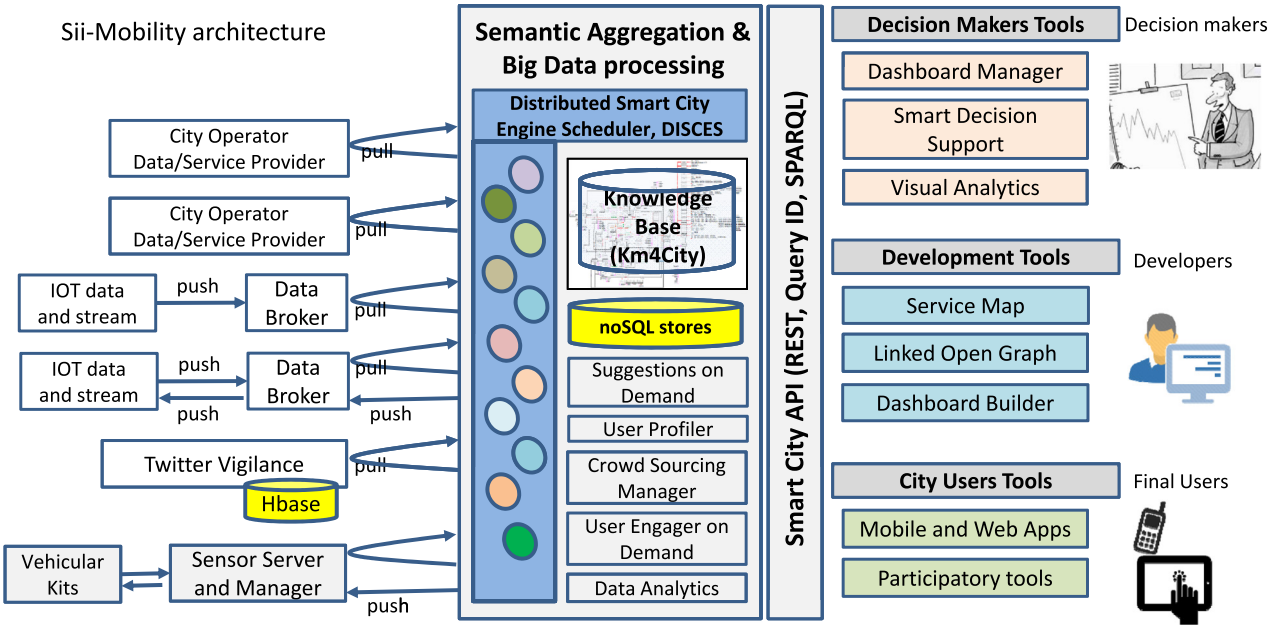


Fig. 4. Sii-Mobility architecture for smart city.

LAMMA weather agency, ARPAT environmental agency, etc., and several private data coming from City/Regionals Operators: mobility, energy, health, cultural heritage, services, tourism, wine and food services, education, wellness, environment, civil protection, weather forecast, etc.

- **Sensor Server and Manager** is a specific data broker to collect and manage data coming from Vehicular Kits developed in Sii-Mobility for monitoring and informing car, bus and bike drivers; and mobile devices/Apps. They can be directly used with mobile devices to provide several functionalities of the Sii-Mobility solution such as: recommendations, navigation, assistance, etc.
- **Twitter Vigilance** platform regarding the monitoring of some Twitter.com channels and users related to the mobility and traffic in the city [55].
- **Mobile phones and other devices** connected to Sii-Mobility servers. In this case, the data collected from mobiles are related to the: (i) position, velocity and acceleration of city users; (ii) requests to the Smart City API such as searching queries; and (iii) requests to the Recommendations and Engager on Demand.

Once the data are collected, the back office activates a number of processes for improving data quality, reconciliating data and converting data into triples for the RDF store of the Knowledge Base [26], implemented by using a Virtuoso triple store. DISCES is allocating processes on a number of virtual machines allocated on the cloud according to their schedule, and according to the requests arriving from the Decision Makers, Developers and Data Analytics (typically 3.5–5 thousand of jobs per day, collecting multiple datasets per job, for example all the buses of a line according to DATEX II protocol [19]). The processes for data collection can be scheduled according to several different policies to cope with Open Data (to verify if they change sporadically), quasi real time data (changing a few times per day), and real time data (changing every few seconds, such as the position of the Bus, or the position of the City Users).

For semantic aggregation of data and service, it has been decided to exploit and improve the Km4City Ontology (http://www.km4city.org) [25,26], as the main ontological model. Km4City is modeling multiple domain aspects related to mobility,

services, Wi-Fi, cultural services, energy, structure (streets, civic numbers, green areas, sensors, buses, etc.) and it has been improved in Sii-Mobility by adding more features and details for mobility, transport, sensors, environment, first responder services, etc. According to READY4SmartCities FP7 CSA project of the European Commission [56], Km4City is one of the most comprehensive smart city ontologies in terms of domain coverage as reported in http://smartcity.linkeddata.es/. Where Km4City is the smart city ontology covering the highest number of domains among the 70 smart city ontologies reviewed, since it covers: weather, cultural heritage, smart sensors, public structures, city, parking, services, transportation, events, and geographic locations. Recently, Km4City also addressed health (with pharmacies, hospital and real time data of first aid), environment (with pollution and pollination, in addition to weather forecast), and private mobility with fuel prices.

In the architecture proposed, in addition to the RDF store for the knowledge base, a number of noSQL Stores (namely: HBase and MongoDB) have been adopted for storing tabular data as those arriving from sensors and user profiles, and to make versioning of collected data that have to be passed into the RDF store for reasoning. This approach allows us to have the needed tabular data accessible for Data Analytics processes such as those performed for the: estimations of recommendations, engagements, traffic flow predictions, parking forecast, clustering of sensor data behavior, and anomaly detection. When needed, federated queries can be performed among RDF and tabular stores.

The resulted architecture provided several services via Smart City API to Decision Maker Tools, Development Tools and City Users Tools. The proposed Smart City API extended the classical concept of Smart City API as described in details in the following sections. Among the Decision Maker Tools the Dashboard Builder allows to define graphical dashboards for setting up smart city control rooms (e.g., http://www.km4city.org/?controlRoom#realtimeData), while the Smart Decision Support is a System Thinking collaborative tool for defining Bayesian decision models [53]. The development tools see the ServiceMap [50] which allows visually formalizing queries and generating calls compliant with Km4City Smart City API. These calls are directly sent via email to the developer for shortening the production of web and mobile applications as described in the sequel. The ServiceMap development tool has been used to create a couple of mobile and web applications such as the mobile App "*Florence what where*", available on all the mobile platforms, plus HTML5, and Windows 10 (http://www.km4city.org/?app). The API calls provoked by the usage of mobile Apps and of Dashboards are the main sources of workload for the smart city API, as highlighted in the following.

4. Assessing workload of smart city infrastructure

A smart city infrastructure may be regarded as composed of 3 main parts: (i) the data ingestion subsystem, (ii) the data analytics and big data processes; and (iii) the front end for monitoring smart city status as a support for decision system, and for providing smart city API to mobile App, etc.

4.1. Data ingestion infrastructure

In the data ingestion subsystem, the main costs (computational and communication) are due to the gathering and processing of data for reconciliation of them into data bases for the further elaborations. According to Sii-Mobility Smart City solution a number of different processes are activated, mainly on the mobility side. The data acquisition processes are taking data with PULL approach from corresponding data brokers. The data reported refer to a typical operating period in January 2017, as reported in Table 2.

A Smart City is a live solution, every 1–2 weeks the workload typically changes, since new data arrive, older approaches for data gathering are discharged in favor of new one. Thus, statistical data do not represent the actual consistent condition of the Smart City. According to Table 2, more than 4800 processes per day are executed coming from 330 sources which are not stable over time. Most of the data refer to the whole region of Tuscany, others are specific for some city. The Tuscany regional level includes a population of 3.5 million o inhabitants plus about 2 million of new tourists per month.

In Table 2, the data ingestion and reconciliation processes are described with their main parameters. The whole set of considered processes include reading and processing (detailed locations can be recovered by using the public ServiceMap at http://servicemap.disit.org):

- AVM: real time bus lines (3 of them in this case) in Florence city,
- new events of entertainment and culture in Florence city (some of them, can be daily, others have longer durations). For example a total of 1921 processes where collected in the month.
- parking area status in 65 locations of Tuscany region, in the major cities,
- weather forecasts in 275 subareas covering the whole Tuscany region areas,
- fuel station prices in 1563 stations distributed in the whole Tuscany region,
- status of the first aid emergency areas in 21 hospitals spread in the whole Tuscany region and in particular close to the major cities: Florence, Pisa, Arezzo, Siena, Pistoia, etc., collected by means of 3 processes,
- traffic sensors in 724 points of measures located in the highways and in major cities of the WHOLE Tuscany region,
- TPL for public transportation includes updated paths for each bus service provider (16 different operators for a total of 2016 lines with 37 839 stop/stations for bus, tram, railways, and ferry, with time schedule) covering the whole Tuscany region. Moreover, 38 243 data describing line schedule have been collected in the month, which are a 2390.19 dataset per operator per day.

The smart city, in effects also collects Tweets from social media via Twitter Vigilance tools installed in the datacenter (http://www.disit.org/tv). Typically, an average of 500.000 tweets per day are gathered and analyzed (for detecting the inception of early warning events, also using sentiment analysis) covering several different areas: mobility, transport, city services, health, terrorism, etc. The specific hashtags, citations and searchers adopted for the tweets collection are listed and accessible on http://www.disit.org/tv, for example for the Florence city we have a combination of "Firenze" keyword with specific keys such as: Aereoporto, #cronaca, #cultura, etc.; and specific keys: @comunefi, @firenzedigitale, @firenzeturismo, @HUBFirenze, @ProtCivComuneFi, @tranviafirenze, @UNI_FIRENZE, #unifi, #bargello, @VisitArtFirenze, #pontevecchio, @BandieraiUffizi, @Biblio_ComuneFi, etc.

In Table 2, a number of metrics are reported to characterize each data source area:

- number of processes executed as ETL by DISCES,
- average number of datasets acquired by each process of the area per day (this is due to the fact that the data reported has been derived by the observation in the time frame and making the average per day, per line, etc.),
- average number of kB produced by each process in terms of triples for the RDF store. This measure gives an idea of the volume of the data augmenting the knowledge base,
- number of datasets read and processed per day,

Table 2
Costs and reliability about the ingestion processes for smart city.

Areas	# of processes on DISCES	Average number of datasets acquired by processes per day	Average number of kB in triples produced by process	# of DataSets read per day	kB produced as triples per day	kB produced as triples per dataset	Typical day					Typical week				
							# Failure	# Success	# Totale	Failure %	Success %	# Failure	# Success	# Totale	Failure %	Success %
AVM (3 Lines)	**3**	16.00	561.73	7 344	338 161	35	143	459	602	23.75	76.25	774	3 661	4 435	17.45	82.55
Events	**1**	61.68	209.48	740	2 514	3	0	12	12	0.00	100.00	0	84	84	0.00	100.00
Parking	**23**	4.44	9.61	1 262	2 767	2	4	284	288	1.39	98.61	4	2 012	2 016	0.20	99.80
Weather	**275**	1.00	28.02	2 194	61 564	28	3	2194	2197	0.14	99.86	31	15 338	15 369	0.20	99.80
Fuelstation	**1**	1386.00	12,992.19	1 386	25 984	9	1	1	2	50.00	50.00	1	13	14	7.14	92.86
First aid	**3**	2.33	28.83	665	8 304	12	3	285	288	1.04	98.96	234	1 782	2 016	11.61	88.39
Traffic sensors	**5**	160.00	12.31	229 280	17 695	0	4	1433	1437	0.28	99.72	83	9 994	10 077	0.82	99.18
TPL	**16**	2390.19	72,339.48	38 243	1 157 432	30	0	16	16	0.00	100.00	20	92	112	17.86	82.14
	330			281 114	1 614 421				4842	9.57	90.43			34 123	6.91	93.09

- kB of triples produced per day and per dataset for the area. These features give a measure of the volume of triples for the RDF stores, and for the communicating workload. The latter measure is more precise than counting the number of triples, since each triple may have short as well as very long URIs.

Thus, Table 2 can be read per line, for example: data about parking are managed by 23 processes, executed 12 times per day (typically during day light and not in the night); TPL data are managed by using 16 processes, collecting about 2390 datasets; traffic sensors are managed by using 5 processes with about 160 datasets per process, and are read about every 5 min.

Moreover, Table 2 also reports for each area the percentage of failure and success for each kind of area, in the day and in the week. These data are reported in real time in the smart city Control Room for monitoring the smart city data ingestion subsystem. Thus, it can be observed, that the solutions present the 90.43% of success per day and the 93% of success for the week. A higher reliability may depend on the communication problems that the single data providers may experience in their own network, and thus if they do not collect data or collect them partially, the problem is propagated on the Smart City infrastructure as well. It can be noted that the back office executes about 4800 processes per day, and 34 000 per week by using the DISCES scheduler.

According to the above described workload, Fig. 5 reports the stacked diagram in which the CPU trends of the different Virtual Machine. VM, of the smart city data center is depicted in a typical week (from 19 to 25 of January 2017, specifically). These VMs address the data ingestion in the back office of the smart city. Among them, VM identified such as: m14, m40, m42, m70, m69, m92 are computational nodes, while VM: m138 and m72 are the master of the scheduler and the data repository, respectively. The management of cloud resources can be performed optimizing the workload [57]. The DISCES tool is a distributed scheduler of processes, and includes its own monitor processes.

The computational load is about 8200 MCkps (millions of Clocks per second). The computational workload has been provided in terms of MCkps coherently to what is typically performed on cloud architectures [57], where the CPUs are almost very similar in terms of floating point operation support. For this reason, all the datacenter managers such as VSphere of VMware provide metrics about CPU workload in term of million or thousands of Ckps. Other measures such as Gigaflops are typically used for assessing CPU performance in short term [58], while are not suitable for long term monitoring of cloud workload since none of the monitoring tools (see for example VMware, Nagios) provide the real time workload in terms of Gigaflops on heterogeneous set of hosts. On the other hand, with the aim of passing from MCkps to the available resource con public cloud one could use converting models such as that of Amazon ECU (EC2 Compute Unit). 1 ECU provides the equivalent CPU capacity of a 1.0–1.2 GHz 2007 Opteron or 2007 Xeon processor. This is also the equivalent to an early-2006 1.7 GHz Xeon processor.

From the reported assessment, a simple elaboration can be performed according to the resources needed to manage different smart city infrastructures. Moreover, higher level relevant figures have been identified such as the MCKPDS: "*averaged number of Ckps needed for managing a single dataset*"; which can be defined as the ratio of the "averaged Ckps per day of workload" (i.e., 8200 MCkps) and the "*number of datasets processed per day*" (e.g., 274.229). Thus, we have about MCKPDS = 29.9 kCkps needed for managing a single dataset, allowing us to add computational nodes when more datasets are going to arrive into the infrastructure. This estimation is valid also if there is a different distribution of datasets kinds, since the concept of dataset size we adopted in terms of kB of triples per dataset is "quite" homogeneous, in the range of 3–30 kB of triples per data set. The adoption and the estimation and the usage of Clock per operation is a viable solution for estimating the costs when larger data are addressed, since the ETL processes are at the end very similar each

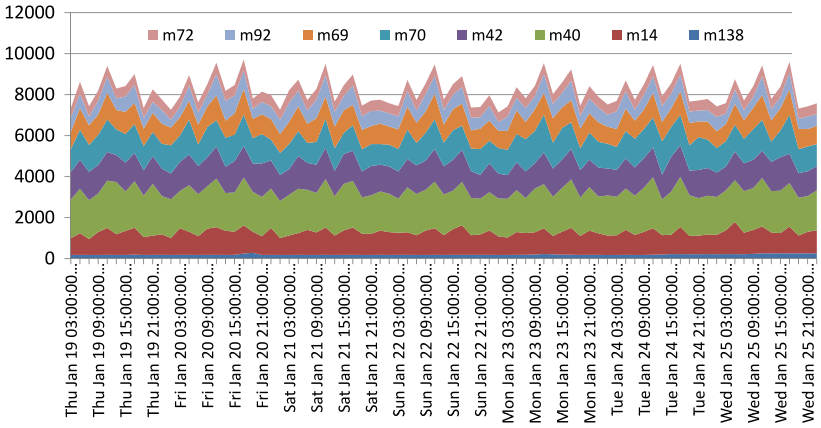


Fig. 5. Stacked workload in terms of MCkps of the VMs allocated for the data ingestion processes.

other. Indeed, a more precise estimation can be performed on the basis of Table 2, which allows understanding how many *Ckps* are needed to process a new bounce of data that produce a number of kB of triples.

According to the above described activity of ingestion and processing, in Fig. 6, the inbound traffic data flow on the network is presented in a stacked manner. Each single VM collects data autonomously from the data brokers (traffic for twitter data are not reported). The needed bandwidth is on average of about 500 kBps. The higher values in some days of the week depend on the weekly processes such as the TPL which may change 1–2 times per week, according to the ordinance and public works in the city. Therefore, the process has to verify every day the presence of new updates (to get the new data for tomorrow), while real updates and thus processing costs and traffic to get massive data are performed and processed 1–2 times per week.

In the data acquisition process, we can have also some streaming data in PUSH. For example, a stream of Wi-Fi data is collected on the same infrastructure, for an average of about 350.000 data packs per day regarding the status of the Wi-Fi, monitoring its usage and track the city users. The Wi-Fi data collection activity has a low CPU cost and thus a maximum of 25 kBps of bandwidth in the peak hours. Moreover, it generates data that needs to be processed as described in the next section.

4.2. Data analytics and big data processes

Once the data are available on the storage, data analytics algorithms can be used to make predictions. The back office for data analytics processes is mainly focused on performing specific elaborations such as:

- Processing of data coming from mobile devices for estimating: typical people and vehicles trajectories by clustering techniques; origin destination matrices; status and the typical travel modality for each user (e.g., car, bus, bike, walking, train); statistical analysis and graphs per user kind, thus for showing to the decision makers about the usage of the city services and mobile App;
- Pre-computing some recommendations per person to be faster in providing them via the mobile App; estimating statistical analysis about the recommendations and how/when they are accepted or not. The precomputation allows us to save time in the final production of recommendations towards the final users on the mobile Apps;
- computing the engagement rules per persons, verifying the single person behavior with the aim of providing engagement messages to the mobile App for implementing some strategies, stimulating a more virtuous behavior for info mobility, sustainability, alerting about weather forecasts, etc. (such as: leaving the private car for public bus, leaving the bus for bike, parking on peripheral zones) [59,60]; performing statistical analysis about the engagements statistics and follow-up. In this activity area some of the computations are performed periodically in background other directly on real time;
- Addressing city users actions on social media by taking data from Twitter, via the Twitter Vigilance tool (http://www.disit.org/tv), thus computing several metrics of volume and performing sentiment analysis, and periodically indexing them for further search and analysis [TwitterVigilance]. The activity aims at providing support for early detection of events, and for assessing the sentiments with respect to the city services;
- Processing Wi-Fi network data for extracting typical city user behaviors (statistics and forecast for the next week and days), typical trajectories, origin destination matrices, etc. [61,62]. This activity aims at knowing and predicting the distribution of city users, for tuning the city services (cleaning, tourism support, police, etc.) and reacting to early inception of critical situations.

The data analytics process has a high reliability of about 99.999% for the execution and completion of the above described tasks. The data analytics areas are addressed by a number of processes, which in some cases have to be synchronized each other and thus have to be carefully scheduled. In particular, about 1400 processes per day and 9800 processes per week are executed. Some of them are just executed once per day (or every two days), others every hour, or much more frequently. For example, the calculation of user behavior for engagement is a performed in quasi real time, being executed 720 times per day; also the Wi-Fi monitoring is performed in quasi real time being executed 288 times per day. This approach allows having a trade off from computational costs and reaction time with respect to the early detection of critical situations in the city (see RESOLUTE H2020 on resilience of transport system http://www.resolute.eu).

In Fig. 7, the stacked diagram of the computational workload in MCkps for the above described processing area is reported. It can be noted that the process assessing the user-behavior (exploited by recommender and engagement data analytics processes) presents the smallest cost among the whole set of process areas. The user-behavior assessment consists of analyzing the collected data about city user behaviors with the aim of identifying preferred POIs (Point of interests), traces (from–to among POIs), kind of behavior (using for example: car, bus, bike, walk, etc.) and corresponding activities. On the other hand, the corresponding computational cost consists is producing the engagements for verifying a combination of conditions for issuing a stimulation and/or information and/or engagement messages. For example:

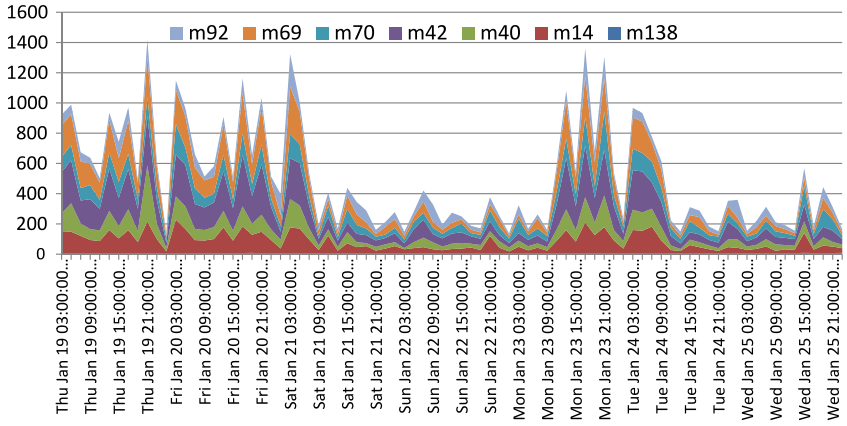


Fig. 6. Stacked network traffic flow for the back office for data ingestion and processing.

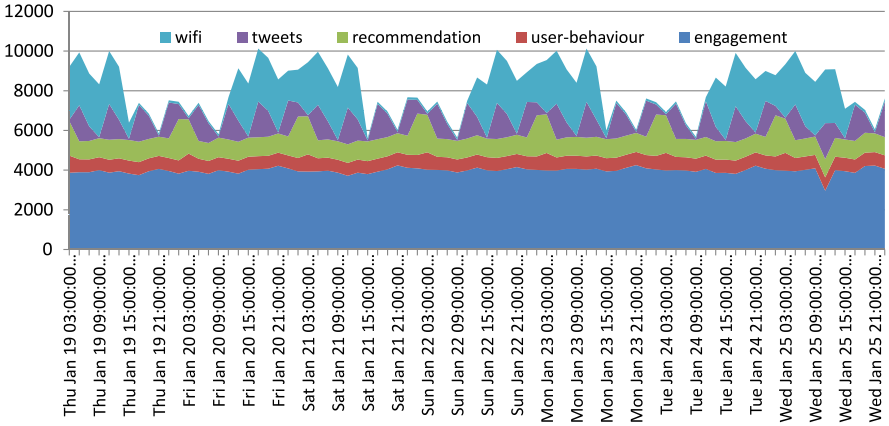


Fig. 7. Stacked computational workload assessed in MCkps about the data analytics back office for smart city.

info-mobility hints, suggesting to take the bus from a bus stop to another instead of using the car, suggest walking if the activity in the day is low, suggest selecting a different parking area, etc. It should be also noticed that, in Fig. 7, we have a relevant graph periodicity in the workload due to the periodic processing of twitter data and the computation of the Wi-Fi analytics for producing assessments and predictions for the decision makers. As a general consideration, also in this case the total computational effort can be allocated on less than 10.000 MCkps. The computational complexity is mainly linear dependent on the population since the most computational expensive processes are depending on that parameter.

Network traffic data are more related to the service requests arriving to the smart city API from the APP and other tools. And thus, they are discussed after to have presented and discussed the Smart City APIs in their complexity.

5. Smart city API of semantic aggregators and reasoners

This section presents the proposed Smart City APIs and compares them with other APIs provided by other solutions belonging to the Case (c) category of data aggregators for Smart City services. The main functionalities of the Smart City APIs for Semantic Aggregators and Reasoners, Case (c), can be classified in few groups that include general functionalities and specific functionalities strictly connected with the domain (search, mobility, environment, user participation and awareness, Smart City interoperability, etc.).

CitySDK, is a *service development kit* for cities and developers that aims to harmonize APIs across cities, [48]. The CitySDK SOA architecture is typically structured on SQL DB. The project makes transformations on each dataset in order to obtain and manage uniform data. There are no semantic relationships among the data collected from data providers; only some of the links between an event and the Point of Interest to which it refers are established. The work on interoperability is limited at the API level, and in the drafting of guidelines and standards for the data providers and the API users. The CitySDK is an open source license free layer, on top of a limited number of applications that are supposed to be provided by third parties or in place in the city.

ECIM has been derived from EPIC [63]http://www.smartmobility.io/, [64], and it has been developed to cover needs of public administrations and small businesses for citizens. The ECIM approach aggregates public and private services to combine them creating and validating new ones. It has been piloted on Brussels, Paris, Barcelona, in the context of an EC project. In this case, there are no semantic descriptors related to the offered services, neither to the datasets. The information coming from data providers are typically converted from their native formats to JSON, and put in a common ECIM MySQL database. Therefore, the ECIM solution differs from the E015 solution [24], since ECIM operates a partial API integration. The offered services can be *either open or at payment*. The ECIM project focuses on mobility, and the participatory actions are planned with the purpose of creating new services meeting citizens' needs.

Transport.API [65] is a new commercial service for providing aggregated open data in the UK. This startup makes available, via a REST API system a relevant number of datasets integrating both static and real-time data, mainly regarding mobility aspects. For these reasons, it can only be partially classified as a Case (c)

solution. Its main limitations are related to the low integration with other domain data, and with the provided licensing model, being a profit service unsuitable for the smart cities.

Navitia.io (https://navitia.opendatasoft.com) is an open source project exploiting OpenDataSoft for open data aggregation in France [12]. The Navitia.io interoperability is at the API level; the datasets are collected from various domains, aggregated into a single database, classified by broad categories (a taxonomy applied on the metadata describing each dataset). There is no uniform semantic model describing and interconnecting the datasets themselves: reconciliation or quality improvement mechanisms are completely missing. For these reasons, it can only be partially classified as a Case (c) solution. Moreover, the service is not hosted on a powerful server, or it is not scalable, thus the users are asked to limit their requests to one per second.

Km4City ontological model and data aggregation of 2014 [25] with its no commercial Smart City API, has been recently augmented with results of the Sii-Mobility project on mobility and transport. Thus, it enables the development and deploy of personal assistant services, for example to implement Mobile Apps for city users with multi-domain information, and at the same time collects data from the users about their preferences and behavior. Km4City can be adopted by city operators and industries to create web and mobile applications, and to exploit published/integrated open and private data, static and real time data. On the other hand, it can be easily extended and applied to different Smart Cities.

API can be classified in those that are accessible for:

- mobile and web Apps, mainly for info mobility and providing services to the city users;
- dashboard panels and Data Analytics processes. They may include classified information, or too technical information to be deployed at the city users;
- higher level control by City Operators (civil protection, major, assessors, etc.), which are totally and strictly classified information such as TV cameras, alarms, etc. (which are not reported in this paper).

In Table 3, the comparison of the above commented Smart City API approaches is presented, based on the main API functionalities and services exposed. From the comparison, it is evident that the major difference has been registered for Km4City/Sii-Mobility that provides support to (i) perform semantic queries with inference on aggregated data, (ii) realize personal assistant, and (iii) implement dashboards.

The proposed approach enabled Sii-Mobility to produce suggestions on demand, as depicted by the workload of the ServiceMap service [50], presented in the next section.

5.1. Structure of smart city API

A strongly relevant difference from the reviewed smart city APIs and those provided by Km4City consists in the possibility of posing requests, by using different modalities:

- **SPARQL Query**: calls are directly performed on the RDF Store endpoint using the standard SPARQL query protocol (based on REST) using GET or POST requests with the *query* parameter containing the SPARQL query;
- **SPARQL Query with Inference**: calls are directly performed on the RDF Store endpoint http://servicemap.disit.org/WebAppGrafo/sparql, using the standard SPARQL query protocol (based on REST) using GET or POST requests with the *query* parameter containing the SPARQL query, including inference aspects in the case of Virtuoso, or automatically exploiting the inference in the case of OWLIM [26]. See Table 3 to see the list of API which exploit the inferences, e.g., API Location, and API GPS;
- **REST**: calls are performed by using APIs using full text, keywords, service ID (URI) to get info, geolocation, service ID (URI) to get closer services, time, etc.;
- **Query ID**: calls are performed by using a QueryID (identification) assigned by the ServiceMap tool manager, after having performed a query by using the graphic user interface, as a visual query.

The following example consists in requesting all services in around 0.1 km with respect to the location of service identified by the URI http://www.disit.org/km4city/resource/45e458d0c9e5bf53e34e7e2fed73a6b4 (which is the Cupola of Santa Maria del Fiore, Florence's Cathedral). The SPARQL query is:

```
SELECT?name?dist WHERE{
?s a km4c:Service.
?s geo:geometry?g1.
?s schema:name?name.
<http://www.disit.org/km4city/resource/45e45 . . . >
geo:geometry?g2.
BIND(bif:st_distance(?g1,?g2) AS?dist)
FILTER(?dist <= 0.1)
}ORDER BY?dist
```

A similar request can be performed using the REST API http://servicemap.disit.org/WebAppGrafo/api/v1/?selection=http://www.disit.org/km4city/resource/45e458d0c9e5bf53e34e7e2fed73a6b4&categories=Service&maxDist=100&format=json, where *selection* indicates the service URI around which the services should be located, *categories* indicates the kind of service to be located, *maxDist* indicates the max distance in meters and *format* indicates the return data format (JSON or HTML). The same query presented above can be performed as Smart City API based on Query ID. This technique assigns a QueryID to a complex query, allowing the developers to copy/paste the REST call, to perform the query identified by the QueryID, disregarding the complexity of the query. This approach simplifies the work of programmers, and at the same time makes the implementation of web and mobile Apps more independent from the data of the smart city. For example, traditionally a certain API call adopted in mobile App can request a list of services' types in an area. If the developer decides to change the list of requested service type, the mobile App has to be rebuilt. On the contrary, adopting an API call based on QueryID in some mobile App deployed in the end of the city users allows avoiding of reinstalling the mobile App to update a query. In fact, the developer can simply change the query associated with the QueryID on the server side without rebuilding the application to see it executed by a mobile App. For example, to get the HTML, the URL represents the same query stored with a QueryID http://servicemap.disit.org/WebAppGrafo/api/v1/?queryId=9e5662a352d90ad4bc77690277a371ab&format=html (see Fig. 8). In Sii-Mobility, the developers may use the ServiceMap tool to visually compose some geographical and textual queries, and from them to request the sending of an email containing the calls that can be performed for obtaining the same resulting data in JSON and/or web page in HTML. In Fig. 8, the ServiceMap at work is presented.

6. Some data results about the usage of Sii-Mobility smart city API

This section reports some data regarding the usage of Sii-Mobility/Km4City Smart City APIs. The Smart City API are exploited by a number of applications: ServiceMap [50], a number of mobile Apps on Google Play, Apple Store and Windows Store (called *Firenze where what. Km4city*, and *Toscana where what.Km4city*), Smart Decision System [53], Florence Dashboard

Table 3
Comparison of smart city API solutions: "(X)" limited or not accessible for all kind of applications, "X1" Open311 Interface, "X2" full text search only on dataset metadata and not on data, "Xi" available feature using a semantic query with inference.

Front end smart city API domains to provide services to management smart city applications, and to web and mobile applications.	CitySDK	ECIM	Transport.API	Navitia.io	Km4City
API: Service Search					
(API GPS) Search Full Text	X2	X2		X	Xi
(API GPS) Search around a GPS point	X	X	X	X	Xi
(API GPS) Get location from GPS					Xi
(API LOCATION) Search along a line, a polyline	X				Xi
(API LOCATION) Search in an area, a closed shape	X			X	Xi
(API LOCATION) Search for street, region, municipality, etc.				X	Xi
API: Mobility					
Get Public Transport, bus-stops, lines, and schedule			X		X
Get Real time delay of Public Buses	X		X	X	X
Get Traffic Flows Status	X	X	X	X	(X)
Get Parking Status	X	X		X	X
Get fuel station status-prices					X
Get ped and vehicle Routing (Latitude, Longitude | POI; Latitude, Longitude | POI)	(X)	(X)	(X)		X
Get an Intermodal Routing	X	X	X	X	(X)
Get an Integrated Ticketing		X	X	X	(X)
Get a Routing for Good Delivering (multi stop planning)	X		X		(X)
API: Environment, Sensors and Actuators, IOT, health					
Get health structures (hospital, doctors, etc.)					X
Get First Aid Status					X
Get Weather Forecast	X			X	X
Get Sensor/Actuator Value/Status	X	X		X	X
Get pollution, temperature, pollination, etc.					X
API: User Participation and Awareness					
Get Social Media Monitoring Info			X		X
Save Crowd Sourcing Comments per service	X1		X		X
Save Crowd Sourcing Votes and Media per service	X	X	X		X
Get/Set variable message panel status per location					X
(API EVENT) Get Events in the city/area (today, week, and month)	X				X
API: Personal Assistant (engagement + recommender)					
Save User Profile	(x)	(x)		(x)	X
Get Suggestions on Demand					X
Get soundages, information, engagements					X
Get Civil Protection news in Push					X
Save Mobile Sensors Status	X	X		X	X
API: Domains of Geo Located Services					
(API INFO) Culture and Tourism	X	X			X
(API INFO) Point of Interest	X	X	X	X	X
(API INFO) Mobility and transport, parking, flow	X	X	X	X	X
(API INFO) Education and training	X	X			X
(API INFO) Government and Pub Services	X	X			X
(API INFO) Commerce and Industry	X	X			X
(API INFO) Health and personal	X	X			X
(API INFO) Public Energy, Energy and home	X	X			X
(API INFO) Energy and Mobility (fuel, recharging)	X	X	X		X
API kind of Call (JSON and/or HTML)					
SPARQL Query					X
SPARQL Query with Inference					Xi
REST	X	X	X	X	Xi
Query ID					Xi
API for Non Functional features					
Direct API Authentication	X	X	X	X	X
API Authentication via Social Media		X			(X)
Data Licensing Control	X		X	X	(X)

(http://www.km4city.org/?controlRoom#realtimeData), and Web app (http://www.km4city.org).

The numbers about the received calls from the mobile and web applications to Smart City API are reported in Fig. 9. The distribution of number of smart city API calls puts in evidence the augmentation of activity during the day light with respect to the night time.

According to Table 3 and Fig. 9 the calls are due to the:

- API Info + Events. API INFO provides data about specific information associated with a given service URI. API EVENT provides information about the event in the area and in the time frame: one or more days, weeks or months (for example in the case of expositions), exploiting temporal reasoning. These kind of API calls are provoked by the user requesting specific information for a service, since he/she is browsing the area, or is requesting to see more details from suggested service of

Fig. 8. Example of ServiceMap results performing the visual query on knowledge base.

which the application has already obtained the URI (so that no inference is exploited in this case). API event provides

- API GPS provide all services of interest with respect to a GPS position according to the user profile, and to the classification provided in the ontology. This API family exploits inference in the knowledge base and it is used as a first query to identify the set of identifiers around a certain point. Then specific API INFO calls are invoked to get details information. In this case, the API exploits the ontology and inference for the classification hierarchy and for the spatial reasoning around a point.
- API LOCATION call that passing GPS coordinates provides back the address (street, civic number, municipality name and URI, etc.). This API family exploits inference in the knowledge base and it is used as a first query for searching geolocated information such as: weather forecast, POS services, mobility services, etc. Moreover, the family of these API exploit the spatial queries that take into account the shape of the municipality, and street graph. In this case, the request exploits the ontology and inference for the classification hierarchy and for the spatial reasoning along a polyline, such as the bus line or a street.
- Recommend Calls are those specifically addressing additional recommendations and not those provided in push.
- ServiceMap Calls and SPARQL query are performed by the developers and/or from Dashboards to analyze the geodistribution of services (http://servicemap.km4city.org) and to perform specific analysis on the knowledge base.

According to Fig. 9, a relevant number of API calls exploit inference on the Knowledge Base located on the RDF store modeled according to the Km4City ontology. See Smart City APIs marked as "Xi" in Table 3 for proposed solution. The calls exploiting the inference are typically those used for the first call to the knowledge base allowing the identification of services. Then specific direct calls for each service are performed to get more information. This fact demonstrated the exploitation of the kB in the design of Case (c) aggregation solution and Smart City APIs. Comparing Figs. 9 and 7, it can be noticed that the workload of the computational activities of data analytics provides a very low correlation with respect to the front-end calls since the two issues are mainly decoupled by the presence of caches from: recommender and knowledge base, engager and knowledge base, and from the smart city API and mobile and web Apps (see Fig. 10).

7. Conclusions

Most of the smart city solutions are transforming data to services for city users and operators, exploiting data and data analytics solutions. Smart City services usually integrate open and private data, static and real time data coming from the administrations and from private operators. Smart City APIs may offer different functionalities depending on the chosen architectural solutions to pass from data to services, limiting or enabling the possibility of exploiting aggregated and reconciliated data by reasoning algorithms enabling the production of sophisticated services, such as those for implementing personal assistants, connected drive, smart services, etc. The effective exploitation of data and semantic relationships to provide smart services stimulated us on creating a smart city architecture with a data aggregation based on semantic computing at the ground. Therefore, the proposed architecture included: (i) a data aggregation layer focused on bringing data into a knowledge base for the city (a sort of expert system with inference capability), (ii) a solution for executing a large range of different data analytics with the support of process management tools, (iii) the formalization of Smart City APIs by which all the web and mobile Apps, and dashboards may access to the data and knowledge base with spatial and temporal reasoning and inference. To this aim, the paper presented a comparison among a range of state of the art architectural solutions for data aggregation for smart cities and for providing Smart City API, offering the comparison in term of requirements coverage and flexibility. The comparisons have been performed to analyze pros and cons of the introduction of semantic computing into the smart city architecture with the aim of providing smart services. The analysis has highlighted

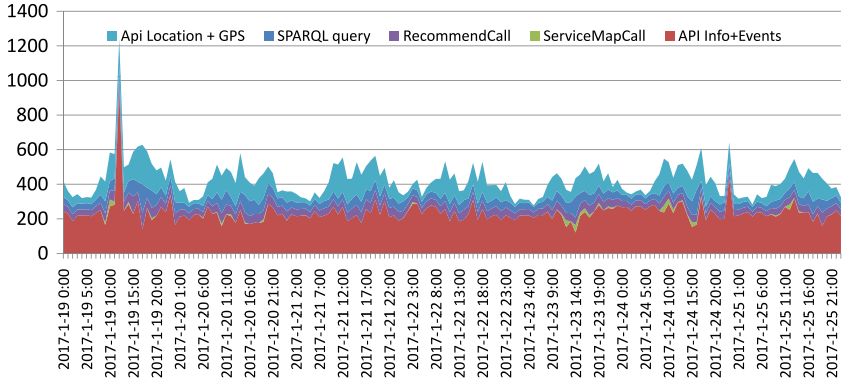


Fig. 9. Stacked diagram reporting the trend of the number of smart city API calls per hour in the same week analyzed in Section 4.

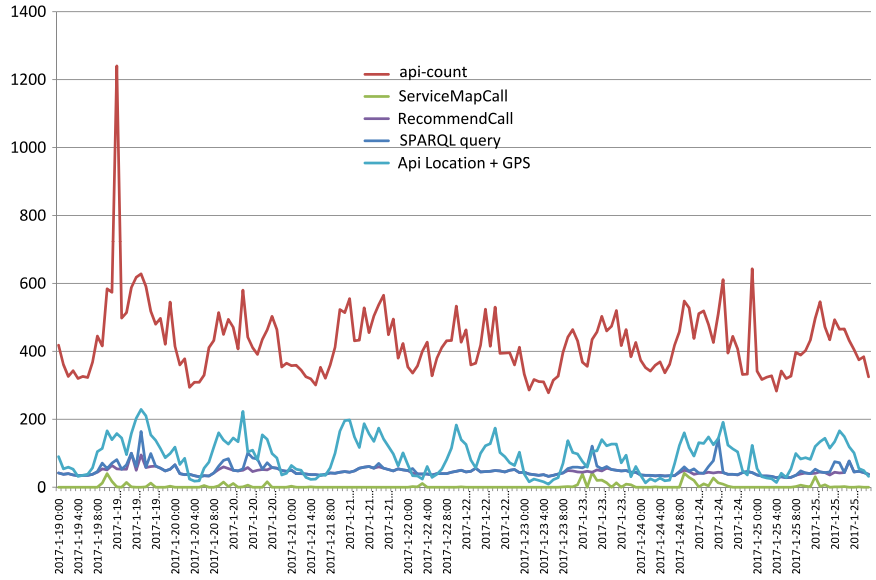


Fig. 10. Trend of the smart city API calls per hour in the same week analyzed in Section 4.

the advantages of presenting semantic interoperable aggregated data which enables to perform spatial, temporal and conceptual reasoning directly exploited by the smart city APIs. Thus, Sii-Mobility smart city architecture and solution has been presented where the knowledge base has been developed by exploiting Km4City ontology.

The proposed solution has been also assessed in terms of computational and network costs on cloud, putting in evidence the back-end costs and those that depend on the front-end activities and requests, and how these two areas can be decoupled. The assessment has been performed by using cloud based metrics and resources which may be accessible on a private cloud that may allow comparing the solution with other solutions. The assessment has been also performed to identify some drivers that may be used to guess the general costs when the solution needs to address more data. Some of the identified drivers for the assessment may allow estimating the needed resources to cope with a range of different smart cities (taking into account, costs in the data ingestion, data analytics and front-end, API).

Sii-Mobility and Km4City smart city tools and solutions are accessible as Affero GPL open sources on Github https://github.com/disit. They are presently adopted by RESOLUTE H2020 and REPLICATE, H2020 European Commission research and development projects and by Ghost of MIUR project on Cagliari Metro Area. Those projects are also contributing on specific topics and interoperable tools in the general architecture.

Acknowledgments

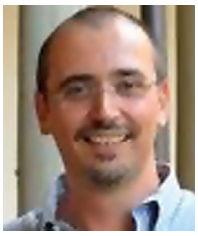
The authors would like to thanks the MIUR Smart City national founding SCN_00112 and to the University of Florence and companies involved for co-founding. Km4City is an open technology of research of DISIT Lab.

References

[1] IBM Smart City http://www-935.ibm.com/services/us/gbs/bus/html/smarter-cities.html.
[2] IBM Institute for Business Value, How Smart is your city? Helping cities measure progress, [online]. Available: oct 2013. http://www.ibm.com/smarterplanet/global/files/uk__en_uk__cities__ibm_sp_pov_smartcity.pdf.
[3] Alcatel-Lucent Market and Consumer Insight team, "Getting Smart about Smart Cities Understanding the market opportunity in the cities of tomorrow", Oct. 2013.
[4] CISCO: http://www.cisco.com/c/dam/en_us/about/ac79/docs/ps/motm/Smart-City-Framework.pdf, http://www.cisco.com, http://www.cisco.com/c/dam/en_us/solutions/industries/docs/gov/everything-for-cities.pdf.
[5] L. Anthopoulos, P. Fitsilis, Exploring architectural and organizational features in smart cities, in: 2014 16th Int. Conference on Advanced Communication Technology, ICACT, IEEE, 2014.
[6] L. Filipponi, A. Vitaletti, G. Landi, V. Memeo, G. Laura, P. Pucci, Smart City: An Event Driven Architecture for Monitoring Public Spaces with Heterogeneous Sensors, in: Sensor Technologies and Applications, SENSORCOMM, 2010 Fourth International Conference on, vol., no., 18–25 July 2010, pp. 281–286.
[7] A. Domingo, B. Bellalta, M. Palacin, M. Oliver, E. Almirall, Public open sensor data: Revolutionizing smart cities, in: Technology and Society Magazine, Vol. 32, IEEE, 2013, pp. 50–56. http://dx.doi.org/10.1109/MTS.2013.2286421, no.4, winter.
[8] Hafedh Chourabi, et al., Understanding smart cities: An integrative framework, in: 2012 45th Hawaii International Conference on System Science, HICSS, IEEE, 2012.
[9] Smart Cities: Ranking of European Medium-Sized Cities, Centre of Regional Science, Vienna University of Technology, 2007.
[10] J.M. Shapiro, Smart cities: quality of life, productivity, and the growth effects of human capital, Rev. Econ. Stat. 88 (2) (2006) 324–335.
[11] CKAN: http://ckan.org.
[12] OpenDataSoft: https://www.opendatasoft.com/.
[13] ArcGIS OpenData: http://opendata.arcgis.com/.
[14] 5 Stars Open Data from Tim Barneers Lee. http://www.slideshare.net/TheODINC/tim-bernerslees-5star-open-data-scheme.
[15] RDF https://www.w3.org/RDF/.
[16] SPARQL: https://www.w3.org/TR/rdf-sparql-query/.
[17] P. Bellini, P. Nesi, A. Venturi, Linked Open Graph: browsing multiple SPARQL entry points to build your own LOD views, Int. J. Visual Lang. Comput. (2014) http://dx.doi.org/10.1016/j.jvlc.2014.10.003, http://log.disit.org.
[18] C. Badii, P. Bellini, D. Cenni, G. Martelli, P. Nesi, M. Paolucci, Km4City Smart City API: an integrated support for mobility services, in: (SMARTCOMP) IEEE International Conference on Smart Computing, IEEE, 2016.
[19] DATEX II: http://www.datex2.eu/sites/www.datex2.eu/files/Datex_Brochure_2011.pdf.
[20] IETF: https://www.ietf.org.
[21] F.J. Lin, Y. Ren, E. Cerritos, A feasibility study on developing IoT/M2M applications over ETSI M2M architecture, in: 2013 International Conference on Parallel and Distributed Systems, ICPADS, IEEE, 2013.
[22] J. Swetina, et al., Toward a standardized common M2M service layer platform: Introduction to oneM2M, IEEE Wirel. Commun. 21 (3) (2014) 20–26.
[23] Green Button Connect: http://www.greenbuttonconnect.com/.
[24] E015 digital ecosystem. http://www.e015.expo2015.org/.
[25] P. Bellini, M. Benigni, R. Billero, P. Nesi, N. Rauch, Km4City ontology building vs. data harvesting and cleaning for smart-city services, Int. J. Visual Lang. Comput. (2014) http://dx.doi.org/10.1016/j.jvlc.2014.10.023.
[26] P. Bellini, I. Bruno, P. Nesi, N. Rauch, Graph Databases Methodology and Tool Supporting Index/Store Versioning, J. Visual Lang. Comput. (2015) http://www.sciencedirect.com/science/article/pii/S1045926X15000750.
[27] N. Korn, C. Oppenheim, Licensing Open Data: A Practical Guide. In: Discovery [online]. June 2011 [cit. 2012-02-20]. Retrieved from http://discovery.ac.uk/files/pdf/Licensing_Open_Data_A_Practical_Guide.pdf.
[28] S. Villata, N. Delaforge, F. Gandon, A. Gyrard, An Access Control Model for Linked Data, in: OTM Workshops, in: LNCS, vol. 7046, Springer, Heraklion, Greece, 2011, pp. 454–463. Oct.

[29] P. Bellini, L. Bertocci, F. Betti, P. Nesi, Rights enforcement and licensing understanding for RDF stores aggregating open and private data sets, in: second IEEE International Smart Cities Conference, ISC2 2016, Trento, Italy, SLIDES, 12 to 15 September 2016. http://events.unitn.it/en/isc2-2016.
[30] S.M. Odeh, A. Mora-Garcia, M.N. Moreno, J.J. Merelo, A Hybrid Fuzzy Genetic Algorithm for an Adaptive Traffic Signal System, Adv. Fuzzy Syst. (2015) 378156:1–378156:11.
[31] P.A. Castillo, A. Fernández-Ares, P. García-Fernández, P. García-Sánchez, M.G. Arenas, A.M. Mora, G. Romero, V.M. Rivas, J.J. Asensio, J.J. Merelo, Studying individualized transit indicators using a new low-cost information system, in: Handbook of Research on Embedded System Design, IGI Global, 2014, pp. 388–407. http://dx.doi.org/10.4018/978-1-4666-6194-3.ch016, (Chapter: 16), January, Editors.
[32] J. Weppner, P. Lukowicz, Bluetooth based collaborative crowd density estimation with mobile phones, in: Proceedings of the 2013 IEEE International Conference on Pervasive Computing and Communications, PerCom, IEEE, 2013, pp. 193–200. http://dx.doi.org/10.1109/PerCom.2013.6526732, March.
[33] F. Mendez, System and method for monitoring people and/or vehicles in urban environments. US Patent Application No: 2011/0128,127. Serial no 13/56,068, 2013. Retrieved from http://www.google.com/patents/US20110128127.
[34] M. Martchouk, F. Mannering, D. Bullock, Analysis of freeway travel time variability using Bluetooth detection, J. Transp. Eng. 137 (10) (2010) 697–704. http://dx.doi.org/10.1061/(ASCE)TE.1943-5436.0000253.
[35] A. Haghani, M. Hamedi, K. Sadabadi, S. Young, P. Tarnoff, Data collection of freeway travel time ground truth with Bluetooth sensors, Transp. Res. Rec. J. Transp. Res. Board 2160 (2010) 60–68. http://dx.doi.org/10.3141/2160-07.
[36] http://www.sii-mobility.org/images/deliverables/Sii-Mobility-DE1-1a-analisi-requisiti-casi-uso-v3-0-pub-final.pdf.
[37] P. Bellini, M. DiClaudio, P. Nesi, N. Rauch, Tassonomy and review of big data solutions navigation, in: Rajendra Akerkar (Ed.), Big Data Computing, Western Norway Research Institute, Chapman and Hall/CRC Press, Norway, ISBN: 978-1-46-657837-1, 2013, pp. 57–101. http://dx.doi.org/10.1201/b16014-4, as (Chapter 2), eBook: 978-1-46-657838-8, july 2013.
[38] G.P. Hancke, B.C. de Silva, G.P. Hancke Jr., The role of advanced sensing in smart cities, Sens. (Switzerland) 13 (1) (2013) 393–425. http://dx.doi.org/10.3390/s1300100393, PMID:23271603.
[39] Specification For Urban Platforms, EIP Project, version 2.2, 2016, European Innovation Partnership for Smart Cities & Communities (EIP_SCC).
[40] A. Cox, P. Parslow, B. De Lathouwer, E. Klien, B. Kempen, J. Lonien, D4.2 – Definition of Smart City Reference Architecture, ESPRESSO systEmic Standardisation apPRoach to Empower Smart citieS and cOmmunities, 2016.
[41] SOCRATA: https://www.socrata.com/.
[42] Bin Cheng, et al., Building a Big Data Platform for Smart Cities: Experience and Lessons from Santander, in: 2015 IEEE International Congress on Big Data, IEEE, 2015.
[43] L. Sanchez, L. Muñoz, J.A. Galache, P. Sotres, J.R. Santana, V. Gutierrez, R. Ramdhany, A. Gluhak, S. Krco, E. Theodoridis, D. Pfisterer, SmartSantander: IoT experimentation over a smart city testbed, Comput. Netw. 61 (2014) 217–238.
[44] K. Zaheer, S. Liaquat Kiani, A cloud-based architecture for citizen services in smart cities, in: Proceedings of the 2012 IEEE/ACM Fifth International Conference on Utility and Cloud Computing, IEEE Computer Society, 2012.
[45] C. Kyriazopoulou, Smart city technologies and architectures: A literature review, in: 2015 International Conference on Smart Cities and Green ICT Systems, SMARTGREENS, IEEE, 2015.
[46] FIWARE solution stack. https://fiware-iot-stack.readthedocs.io/en/latest/.
[47] S. Kotoulas, et al., Spud—semantic processing of urban data, Web Semant.: Sci. Serv. Agents World Wide Web 24 (2014) 11–17.
[48] CitySDK: http://www.citysdk.eu.
[49] OASC: http://oascities.org/ NGSI OASC: http://wiki.fiware.org/FIWARE_NGSI_Open_RESTful_API_Specification.
[50] ServiceMap: http://servicemap.disit.org, http://www.disit.org/6597.
[51] F. Scharffe, et al. Enabling linked-data publication with the datalift platform, in: Proc. AAAI workshop on semantic cities. 2012.
[52] C. Bizer, J. Volz, G. Kobilarov, M. Gaedke, Silk - a link discovery framework for the web of data, in: CEUR Workshop Proceedings, vol. 538, 2009, pp. 1–6.
[53] M. Bartolozzi, P. Bellini, P. Nesi, G. Pantaleo, L. Santi, A smart decision support system for smart city, in: IEEE International Conference on Smart City/SocialCom/ SustainCom together with DataCom 2015 and SC2, IEEE press, Cina, 2015, December.
[54] P. Bellini, P. Nesi, G. Pantaleo, Benchmarking RDF stores for smart city services, in: 2015 IEEE International Conference on Smart City/SocialCom/SustainCom, SmartCity, IEEE, 2015.
[55] Twitter Vigilance. http://www.disit.org/tv.
[56] READY4SmartCities. http://smartcity.linkeddata.es/index.html.
[57] P. Bellini, I. Bruno, D. Cenni, P. Nesi, Managing Cloud via Smart Cloud Engine and Knowledge Base, Future Gener. Comput. Syst. (2016) http://dx.doi.org/10.1016/j.future.2016.10.006, http://www.journals.elsevier.com/future-generation-computer-systems.
[58] R. Prodan, S. Ostermann, A survey and taxonomy of infrastructure as a service and web hosting cloud providers, in: GRID, 2009, pp. 1–10.
[59] J. Lehmann, M. Lalmas, E. Yom-Tov, G. Dupret, Models of User Engagement, in: In User Modeling, Adaptation, and Personalization, in: Lecture Notes in Computer Science, Vol., 2012, pp. 164–175.
[60] P. Nesi, G. Pantaleo, M. Tenti, Geographical localization of web-visible human activities by employing natural language processing, pattern matching and clustering based solutions, J.: Eng. Appl. Artif. Intell. (2016) http://dx.doi.org/10.1016/j.engappai.2016.01.011.
[61] A. Danalet, M. Bierlaire, B. Farooq, Estimating Pedestrian destinations using traces from WiFi infrastructures, in: Pedestrian and Evacuation Dynamics 2012, Springer International Publishing, 2014, pp. 1341–1352.
[62] P. Bellini, D. Cenni, P. Nesi, AP positioning for estimating people flow as origin destination matrix for smart cities, in: The 22nd International Conference on Distributed Multimedia Systems, DMS 2016, Italy, 2016.
[63] ECIM, European Cloud Marketplace for Intelligent mobility. http://ecim-iminds.rhcloud.com, (FP7 CIP-ICT-PSP-2013–2017).
[64] EPIC, European Platform for Intelligent Cities. http://www.epic-cities.eu, ICT PSP (2011-2013).
[65] Transport API. http://www.transportapi.com.

Claudio Badii graduated in Computer Engineering at the University of Florence. He is a research fellow and doctoral student at DISIT Lab (DINFO) of the University of Florence. He worked in the past to develop a system which simulates the behavior of a cloud platform. Currently he focused on the development of mobile applications in the Smart City and Big Data.



Pierfrancesco Bellini has a degree in Computer Engineering and received his Ph.D. in Computer Engineering and Telecommunications at the University of Florence. E'currently professor and researcher at the Department of Information Engineering of the University of Florence and Professor of Operating Systems. His main research interests include semantic computing, ontology engineering and cloud computing.

Daniele Cenni has a degree in Computer Engineering and received his Ph.D. in Telematics and Information Society at the University of Florence. It is Research Fellow at DISIT Lab (DINFO) of the University of Florence. His main research interests include information retrieval and analysis of user behavior in context Social Network and Smart City.

Angelo Difino graduated from the Department of Computer Science of the University of Turin in 2000. He has worked in the past with personalization and multimedia technologies in the MPEG and presently is Ph.D. student at DISIT, laboratory of the University of Florence, on machine learning technologies capable of achieving scenarios assistance/engagement.

Paolo Nesi is the chair of the DISIT Lab at the University of Florence, and Professor of computer science. He has coordinated several research projects and innovation, the European Commission, national and regional. Currently coordinating the research line Km4City and projects Be-Mobility MIUR Smart City National mobility and transport, RESOLUTE H2020 resilience in critical infrastructures and transportation systems, while DISIT lab contributes in various other projects of big data, Smart City, Smart cloud, industry 4.0.

Michela Paolucci received her degree in Telecommunications Engineering in 2007, and the Ph.D. degree in Telematics and Information Society in 2012. Her research studies are related to: study and realization of ontologies and semantic models, study and development of ETL (Extract Transform and Load data) processes, design and implementation of Best Practice Networks; matchmaking, crowdfunding and crowdsourcing activities; social media analysis, user profiling, data modeling, collaborative work, advanced systems for knowledge management, IPR models. She worked in several projects, including: RESOLUTE, REPLICATE, SiiMobility, Coll@bora, Feedback, SACVAR, OSIM, APRETOSCANA, ECLAP, Axmedis, IMAESTRO, Mobile Medicine.