FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

# Data Models for Smart City IoT

**Manuel João Gonçalves Vieira de Castro**


Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Professor Ana Aguiar (PhD)

June 27th, 2017

# Data Models for Smart City IoT

**Manuel João Gonçalves Vieira de Castro**

Mestrado Integrado em Engenharia Informática e Computação

Aprovado em provas públicas pelo Júri:

Presidente: José Magalhães Cruz (Professor)

Vogal Externo: José Maria Fernandes (Professor)

Supervisor: Ana Aguiar (Professor)

_____

June 27th, 2017

# Resumo

Numa era onde as sociedades caminham para soluções mais sustentáveis nas suas infra-estruturas, a Internet of Things está a capacitar o sector das smart cities, desempenhando um papel cada vez mais importante em áreas como a qualidade ambiental, o controlo do tráfego, a segurança e a saúde pública. A Ubiwhere encontra-se neste momento a desenvolver soluções de *software* no âmbito da plataforma de cidades inteligentes Citibrain, responsável pela recolha, processamento, armazenamento e distribuição de informação sensorial e serviços web ligados, na sua maioria, às áreas da mobilidade e qualidade ambiental. No entanto, devido à natureza diversa dos sensores e da informação recolhida, os dados obtidos estão, geralmente, em formatos diferentes, impondo grandes barreiras à interoperabilidade dos dados, dificultando as tarefas de integração de sistemas no desenvolvimento de software. Isto representa um grande problema para quem desenvolve aplicações, devido à grande quantidade de tempo e esforço envolvidos na adaptação das mesmas a estes formatos.

Esta tese propõe uma *framework* para comparar e classificar modelos de dados existentes para a Internet of Things. FIWARE, OGC SensorThings, oneIoTa, CitySDK, W3C Generic Sensor, OData, e IPSO Smart Objects são comparados com o objectivo de determinar a melhor alternativa para as soluções Citibrain. É feita uma caracterização dos modelos relativamente aos sectores das smart cities aos quais se aplicam e ao seu nível de abstracção. Este estudo preliminar é concluído com a seleção de modelos capazes de representar toda informação relativa às soluções Citibrain – FIWARE, SensorThings, Generic Sensor, oneIoTa e IPSO Smart Obejcts.

Numa fase seguinte, um conjunto de criterios é selecionado para uma avaliação mais profunda dos modelos de dados, usando métricas como os *overheads* introduzidos tanto no processamento dos dados como na comunicação do sistema, a quantidade e qualidade da documentação existente para cada modelo, assim como a facilidade de implementação. Para o cálculo destas métricas, é criado um tradutor semântico capaz de transformar os dados Citibrain em qualquer um dos modelos estudados em pormenor. Para concluir a análise, uma fórmula pesada é criada, capaz de traduzir a avaliação feita num único valor, para uma fácil comparação dos modelos. O método desenvolvido é então aplicado aos cinco modelos seleccionados, com o intuito de eleger a melhor solução para a Citibrain.

Aplicando os pesos que melhor expressam os requisitos da Citibrain, Fiware atingiu os melhores resultados e foi seleccionado para a implementação de uma aplicação *dashboard* com uma arquitectura cliente-servidor. Esta aplicação serve como uma prova de conceito do estudo efectuado e é responsável por mostrar ao seu utilizador informação real recolhida pelos sensores Citibrain.

# Abstract

In a time where societies are committed to move towards more sustainable solutions in what concerns their infrastructures, the Internet of Things is empowering the smart city sector, playing an increasingly important role in areas such as environment quality, mobility, security and public health. Ubiwhere is currently partaking in the development of Citibrain, a smart cities platform, responsible for the gathering, processing, storage and distribution of sensorial information and web services related, in their majority, to the fields of mobility and environment quality. However, due to the diverse nature of the sensors and the collected information, the obtained data is conveyed in various formats, posing great barriers to data interoperability. This represents a problem for developers, due to the amount of time and effort needed to adapt applications to this heterogeneous data.

This thesis proposes a framework to compare and rank existing data models for the Internet of Things in the smart city sector. FIWARE, SensorThings, CitySDK, oneIoTa, OData, W3C Generic Sensor, and IPSO Smart Objects are compared to determine which one is the best fit for Citibrain's solutions. A characterization of the data models is performed, regarding the smart city sectors they apply to, and their level of abstraction. This preliminary study is then concluded with the selection of the models capable of depicting all Citibrain's solutions' data – FIWARE, SensorThings, oneIoTa, W3C Generic Sensor and IPSO Smart Objects.

In the following stage, a set of criteria is selected for a deeper evaluation of the data models, making use of metrics such as the overheads introduced in both the data processing and communication aspects of the system, the amount and quality of available documentation and support, and the models' easiness of implementation. For the calculation of the metrics covered by the framework, a semantic translator is created, capable of converting all Citibrain's data into the studied data models. To conclude the analysis, a weighted formula is devised, capable of translating the performed evaluation into a single value, for an easier comparison of the models. The five selected models are then put through the developed evaluation process, in order to elect the *de facto* best solution for Citibrain.

Applying the weights that express Citibrain's requirements, FIWARE achieved the best results in the analysis, and was selected for the database implementation of a client-server dashboard application. The dashboard served as a proof-of-concept, and is responsible for displaying real information gathered from Citibrain's sensors.

# Acknowledgements

This dissertation marks the culmination of my academic journey. As such, I would to like to thank all the people that somehow influenced me and fomented my personal growth through these last years.

Firstly to my supervisor, Professor Ana Aguiar, for the guidance and invaluable insights throughout this project. Secondly to my co-supervisor, Ricardo Vitorino, for the opportunity to develop this project, and all the expertise passed onto me. I would also like to say a big thank you to the whole Ubiwhere team, for making me feel at home, and specially to João Garcia and Pedro Diogo for the constant promptitude for helping me with the more technical aspects of the project.

To all my friends, new and old, as well as my professors and classmates at FEUP, I express my gratitude for the companionship during these years.

Lastly, but most importantly, to my family, and, more specifically, to my parents and my sister for the support throughout the entirety of this journey, and for always being by my side.

João Vieira de Castro

“Free time is the enemy of progress.”
Casey Neistat

# Contents

# List of Figures

# List of Tables

# Abbreviations

| | |
|---|---|
| CoAP | Constrained Application Protocol |
| CRUD | Create, Read, Update and Delete |
| HTTP | Hypertext Transfer Protocol |
| ICT | Information and Communication Technology |
| IoT | Internet of Things |
| IDE | Integrated Development Environment |
| IPv6 | Internet Protocol version 6 |
| JSON | JavaScript Object Notation |
| LOD | Linked Open Data |
| LOV | Linked Open Vocabularies |
| LWM2M | Lightweight Machine-to-Machine |
| OMA | Open Mobile Alliance |
| OWL | Web Ontology Language |
| POI | Points of Interest |
| RAML | Restful API Modeling Language |
| RDF | Resource Description Framework |
| REST | Representation State Transfer |
| SWoT | Semantic Web of Things |
| URI | Universal Resource Identifier |
| WoT | Web of Things |
| XML | eXtensible Markup Language |

# Chapter 1

# Introduction

Created in the beginning of the 21st century, the concept of smart city, "a city in which information and communication technologies (ICT) is merged with traditional infrastructures, coordinated and integrated using new digital technologies" [1], has been in constant refinement. With recent improvements in ICT, resulting in ubiquitous Internet access and lower prices in mobile devices and sensors, the Internet of Things (IoT) became an ever-increasing market, tightly linked with the empowerment of the smart city sector [2].

However, due to the vastness and heterogeneous nature of the data obtained from the many activities taking place in a city, creating smart city solutions can be a rather complex process. Combining the diverse nature of the data with the diversity of data providers, developers who constantly need to adapt their solutions in order to be compliant with both these formats and the ever increasing number of protocols, standards and middleware platforms, face challenges of great magnitude.

This work focuses on the study of data models for the IoT as a solution for the standardization of the information provided by different sensors measuring data of different nature. Such standardization serves as a means to achieve the much needed data interoperability among smart city IoT platforms.

## 1.1 Context

This dissertation is conducted in partnership with Ubiwhere[1], a Portuguese software company based in Aveiro, Portugal. Ubiwhere's business activity focuses on four main sectors, (i) Telecommunications and Future Internet, (ii) Sustainable and Efficient Resource Management,

[1] www.ubiwhere.com

(iii) Transports, Travel and Tourism, and (iv) Knowledge, Collaboration and Education. More specifically, this project falls under the umbrella of Citibrain, a commercial joint venture composed of Ubiwhere and two other Portuguese information and technology (IT) companies: Micro I/O[2], and Wavecom[3]. Citibrain provides Internet of Things solutions that use the same stack and technological architecture (from now on referred to as vertical solutions) in the domains of mobility and environment quality, with the purpose of converging as a platform for smart cities. Smart Air Quality Management, Smart Waste Management, Smart Parking Management and Smart Traffic Management are the four vertical solutions currently offered by Citibrain, which will be described in detail in subsequent chapters.

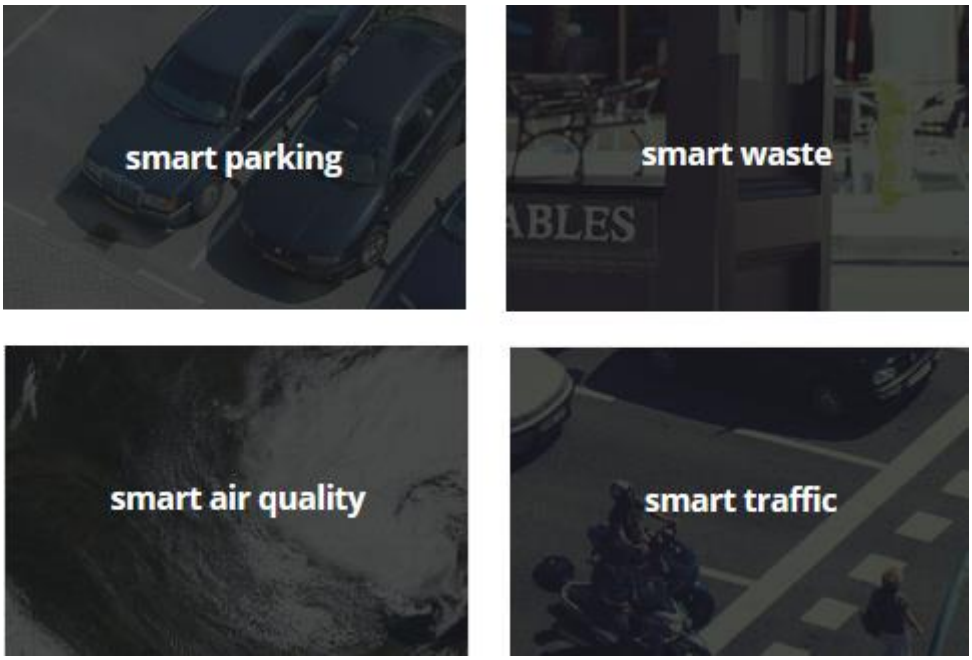



**Figure 1: Citibrain's vertical solutions [3]**

## 1.2 Motivation and Goals

The motivation behind this study arises from the data heterogeneity problem present across current IoT systems. Due to the diversity of the sensed information and the diverse formats it is conveyed in, coupled with the vastness of existing standards and protocols, there are great efforts and time spent by the developers in order to manually adapt their applications to these incongruous formats and to ensure compatibility across systems [4]. Thus, to contribute to overpass this situation represents a meaningful challenge.

[2] www.microio.pt
[3] www.wavecom.pt

One solution for the mentioned problem is the standardization of the way different systems describe knowledge about the domains they focus their activity in. This dimension of interoperability can be achieved with the use of data models - information structures whose main goal is to organize data elements and standardize the way they relate to their properties and with each other.

When applied to the area of the IoT, data models allow many-to-many interactions between applications and things (devices), provide standard interfaces for things and software, and also structure data to enable reasoning approaches to deduce new knowledge [5].

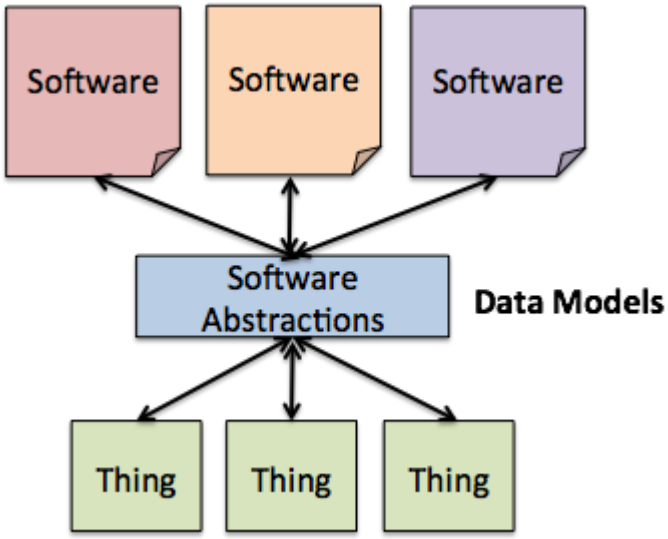



**Figure 2: Data Models and IoT Middleware** [6]

With this motivation in mind, this project will consist of a study of data models for the IoT as a means to achieve semantic interoperability – the ability of systems to communicate and exchange data with unambiguous, shared meaning [7]. The bulk of the work will consist in (i) a research of currently available data models for smart city IoT, (ii) the identification and selection of valid criteria for the evaluation of said models, and (iii) the definition of a formula capable of translating all these metrics into a single value, providing an effective methodology for the quantification of a data model's suitability for an IoT platform. This knowledge will then be used for the evaluation of the discovered data models, with the aim of finding the one that best suits Citibrain's platform.

## 1.3 Contributions

The main contributions of this dissertation include: (i) a description and characterization of a set of data models, (ii) the definition of valid criteria for the evaluation of data models, (iii) the creation of a formula capable of determining the selected data models' suitability to Citibrain's

solutions, based on the chosen criteria in the previous point, and (iv) the development of a dashboard application to validate the study, using the most suitable data model for the server communication.

## 1.4 Dissertation Structure

The current report, after this Introduction, includes a Chapter 2, consisting of an analysis of the state of the art of data interoperability in smart cities, detailing (i) the different challenges of data interoperability in IoT systems, (iii) the semantic web approach to data interoperability, and (iii) the process of evaluating data models, including a study on the quantification of qualitative criteria.

Chapter 3 contains a deeper explanation of the problem and its solution, detailing Citibrain's solutions' architecture and requirements definition, the methodology used throughout the development of the project, the solution's architecture, a description of the developed proof-of-concept client, and the identification of the data model evaluation phases.

In Chapter 4 a description of the seven discovered data models for smart city IoT is presented, namely FIWARE, OCG SensorThings, CitySDK, OData, W3C Generic Sensors, oneIoTa and IPSO Smart Objects, culminating in the selection of five models for a further study.

Chapter 5 presents a description of the developed data model comparison framework, composed of the selection of data model evaluation metrics, the creation of a formula, translating the criteria into a value, as well as some considerations regarding the devised framework.

Chapter 6 contains an in-depth analysis of the five selected data models in the study made in chapter 4, with the purpose of calculating a data model's suitability for Citibrain's vertical solutions. Both a qualitative and a quantitative analysis of the data models are performed, and the chapter is concluded by the selection of a data model to be used by Citibrain, and in the development dashboard application that serves as a proof-of-concept of the performed study.

Finally, Chapter 7 contains the conclusions of the dissertation, including some closing remarks on the achievement of the proposed goals, the main contributions of this thesis, the limitations of the performed analysis, as well as a statement for future work.

# Chapter 2

# The IoT and Smart City Semantics

This chapter reviews previous work in the field of the IoT, related to its smart city sector and quality evaluation of data models.

In a first step, a study in the field of data interoperability is performed, including a description of its four main challenges, as well as the added value purveyed by semantic interoperability. Then, a brief description of the semantic web is carried out, mentioning the importance of two of its technologies, namely Linked Data and Ontologies, and how they apply to the IoT. Finally, an analysis of data model evaluation techniques is carried out, collecting metrics of already defined evaluation frameworks, and also addressing the issue of quantification methods of qualitative criteria

## 2.1 Data Interoperability in IoT Systems

In their 2015 paper, Serrano et al. [8] identify four main challenges regarding data interoperability in IoT systems:

- **Technical Interoperability** – concerning heterogeneous software and hardware, such as communication protocols;
- **Syntactical Interoperability** – concerning both different data formats and different software for the development of ontologies and semantic datasets;
- **Semantic Interoperability** – concerning ontology heterogeneity, differences in terms used to describe data, as well as the meaning of the exchanged data according to the context;

- **Organizational Interoperability** – concerning heterogeneity of different infrastructures.

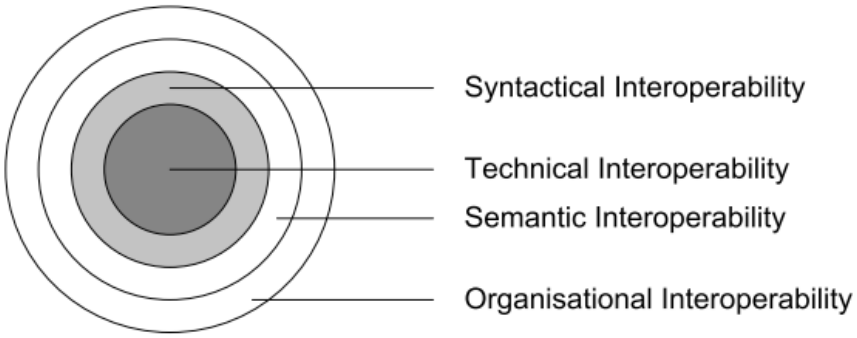



**Figure 3: The four challenges of data interoperability** [9]

According to Consoli et al. [10], the "large, heterogeneous data sources in smart cities make the problem even harder, as different semantic perspectives must be addressed in order to cope with knowledge source conceptualizations". In the same paper, answers to these challenges are pointed out, through the employment of semantic web standards and practices. The authors consider that syntactical interoperability, which regards the used data formats, can be achieved by the adoption of universal knowledge representation languages, such as Resource Description Frameworks (RDF), Web Ontology Languages (OWL) or Linked Open Data (LOD). Besides, it is proposed that semantic interoperability, which regards the meaning of the exchanged data and the different terms used to describe it, may be achieved through the use of a uniform data representation, as well as the formalization of all the concepts into a holistic data model.

## 2.2 The Semantic Web Approach

In recent years, the Internet of Things or, more precisely, its connectivity service - the Web of Things, started integrating semantic web technologies [11]. The Semantic Web, an extension of the Web through standards provided by the World Wide Web Consortium aims at, according to Berners-Lee et al [7], providing structure and meaning to the web pages' content, thus creating an "environment where software agents roaming from page to page can readily carry out sophisticated tasks for users". In the same paper, the authors describe the Semantic Web as an extension of the current Web, where information and data can be processed automatically, with the contents also being destined to be read and comprehended by machines, instead of being only human-readable as is the case with the current Web. By adding logic to the Web, and giving computers access to structured collections of data and sets of inferences rules, computers become able to conduct automated reasoning, to make inferences and to choose courses of action and

answer questions, making use of technologies such as universal knowledge representation languages.

By merging the notions of Semantic Web and the Web of Things, a new concept was formed, the Semantic Web of Things (SWoT). The SWoT envisions the integration into the physical world of information which is simultaneously rich in semantic terms, and easily accessible; it also aims at connecting smart objects and digital entities. According to Jara et al.[11], the SWoT's purpose is the enablement of "knowledge-based systems to achieve high degrees of autonomous capability for information storage, management, and discovery, therefore providing transparent access to information sources", in a reality dominated by constrained devices, with low memory capacity, little processing capabilities and low throughput wireless links [7].

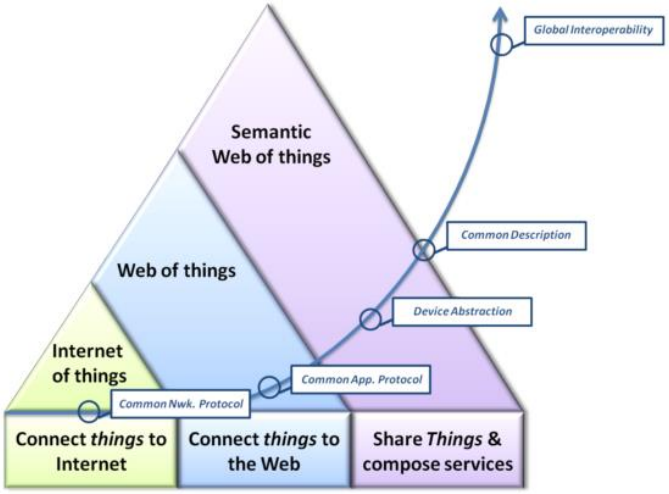



**Figure 4: Evolution of the market size from the Internet of Things to the Semantic Web of Things** [11]

Jara et al. [11] identified the three main interoperability challenges to be solved in order to complete the transition from the IoT to the SWoT:

- Heterogeneous device integration;
- Device abstraction;
- Syntactic and semantic interoperability.

In the same paper, the authors also address possible solutions for the challenges. For the support of heterogeneous devices, the IoT contains a common addressing space (such as IPv6, gateways or middleware), which allows the connection of all types of smart devices and things to the web in a homogeneous way.

By using web technologies, such as RESTful architecture, that define a communication layer where the resources are identified by a Universal Resource Identifier (URI), device abstraction can be attained. These URIs, besides providing abstraction, according to Jara et al. [11], are also able to "define semantic descriptions following structures such as Web Linking".

Regarding syntactic and semantic interoperability, two semantic web technologies have been playing an important role in the SWoT: Linked Data and Ontologies, which will be described in the next subsections.

### 2.2.1 Linked Data

According to Berners-Lee [7], for the Semantic Web to properly function, "computers must have access to structured collections of information and sets of inference rules that they can use to conduct automated reasoning". Linked data consists of, according to Bizer et al. [12], a "set of best practices for publishing and connecting structured data on the Web". In the same paper, the authors also note that while, traditionally, data published on the web consists of dumps of raw data in formats such as CSV, XML or HTML tables, sacrificing a great part of its structure and semantics, there has been a progression of the Web into a new space. This new version of the web can be defined as a space where not only the documents are connected through hyperlinks, but also as a space where the information, concepts and definitions contained in them are also linked, becoming sharable, extensible and easily re-usable.

The concept of Linked Data saw its inception in 2006, when Tim Berners-Lee published a set of four rules for "publishing data on the web, in a way that all published data becomes part of a single, global, data space" [12][13]:

1. Use URIs as names for things;
2. Use HTTP URIs so that people can look up those names;
3. When someone looks up a URI, provide useful information, using the standards (RDF, SPARQL);
4. Include links to other URIs so that they can discover more things.

These basic rules were named Linked Data Principles, and provide means for data publication on the Web and to connect it between different data sources, allowing the linkage of data in different sources, with the employment of RDF and the Hypertext Transfer Protocol (HTTP) [14]. Linked Data is becoming an increasingly important part of the current smart city solutions, due to its interoperable nature and means of data representation. Consoli et al. [10], in the case of Catania, were able to develop a single Linked Open Data model, containing all the city's information, regarding its roads and urban transportation, public lighting maintenance

waste management and urban fault reporting, by converting this heterogeneous information into RDF, using mainly TABELS and Java scripts. The model was later complemented with a common ontology, developed in OWL, which described the city's business processes. The result was a semantic model that allowed the use of data extracted from different sources and unified under a shared semantic model, creating a platform for developing applications based on geo-location, real-time road traffic analysis and public transport management.

### 2.2.2 Ontologies

According to the World Wide Web Consortium, regarding the Semantic Web, vocabularies and their more complex formal counterpart, ontologies, "define concepts and relationships used to describe and represent an area of concern" [15].

The main roles of a vocabulary on the Semantic Web are, on one hand, to promote data integration, by resolving ambiguities on the terms of different data sets, and on the other hand, when new knowledge enables the discovery of new relationships [14].

Chandrasekaran et al [16] identify two major ontology classes regarding Knowledge Base Systems:

- **Ontology as vocabulary**, characterizing vocabularies as providing "a set of terms with which to describe the facts in some domain, while the body of knowledge using that vocabulary is a collection of facts about a domain";

- **Ontology as content theory**, characterizing ontologies as content theories, given their contributions to a domain – identifying the sorts of objects, their properties, as well as the relations between them.

In the same paper, the authors classify ontological analysis as a method for the clarification of the structure of knowledge. Within a domain, its ontology provides the central part of the information defined by it, and, without ontologies or the existence of the concepts defined by them, the existence of a vocabulary capable of representing the domain's knowledge is an unviable task.

According to Gyrard et al. [8], the rise of popularity of the ontologies is due to its capacity of easing the interoperability among applications, services, software and platforms and for describing domain knowledge. The authors identify the main benefits of using ontologies as (i) exchanging data between systems, (ii) providing interoperability among systems, (iii) designing and sharing knowledge, and (iv) simplifying operations.

The British Standards Institution developed a smart city concept model [17], able to describe all the entities, actors, occurrences and observations of a city through the use of ontologies. The

resulting model is able to provide complete syntactical interoperability among the sectors of a city, leaving doors open for the goal of semantic interoperability.

Gyrard et al. [18] proposed LOV4IoT, a dataset comprising circa 300 projects based on ontologies, referencing and classifying metrics such as (i) their applicative domains, (ii) the used sensors, (iii) the status and the information on the used ontologies, (iv) the reasoning that allows high level abstraction deductions, and (v) papers associated with the project [19]. Besides the classification of ontologies, the aim of the dataset was to promote the reuse of concepts in the development of new ones, with the final goal of attaining interoperability both among city sectors and among cities. The authors also proposed the Machine-to-Machine Measurement (M3) Framework, responsible for the generation of IoT application templates, based on the users' employed sensors and domains, and based on the ontologies present in LoV4IoT.

### 2.2.3 Empowering Ontologies with Linked Data

While ontologies are able to both provide terms to describe the facts in some domains and characterize their objects, properties, and relationships [16], there is still a lack of consistency between different domains. While the reuse of ontologies, as proved by Gyrard et al. [18], may ease the interoperability of information between systems, such interoperability is achieved by the reuse of already existing ontologies and vocabularies. A wider level of data interoperability may be achieved with the employment of the Linked Data principles, through a process called *Ontology Alignment,* which consists in the creation of relations between entities of different ontologies [20]. With the establishment of such consistency among ontologies in different domains, a new level of data interoperability is created, which can be achieved without sacrificing a system's identity and already defined and implemented concepts and relations.

Parundekar et al.[20] devised an alignment algorithm for ontologies, which uses data analysis and statistical techniques for class-matching among different ontologies. Besides data consistency among systems, the developed algorithm was also able to generate new, more refined ontologies for sources defined with simple ontologies, through the alignment of said systems with systems with more complex ontologies. The authors' algorithm was then tested in Liked Data sources from geospatial, genetics and zoology domains. The methodology was applied to five pairs of data sources, constructed with six different ontologies: (i) DBpedia, (ii) LinkedGeoData, (iii) Geospecies, (iv) Bio2RDF 's MGI, (v) Bio2RDF 's GeneID, and (vi) Geonames. The algorithm was able to discover circa 800 equivalences and 29000 subset relationships among the five pairs of source domains.

## 2.3 Data Model Evaluation

Over the years, the process of evaluating data models has been quite difficult to define and characterize. According to Maier [21], since data modeling has a varying nature, no actual general concept can be given for the evaluation of data models. According to Moody et al. [22], one of the biggest problems presented in the design of data modeling solutions is the substantial alternative designs that can be developed for a given problem. Hence, the process of data modeling should not be seen as a deterministic search process for the "correct data model", but as a "process of searching for alternative solutions. In order to get the best result, it is important to explore alternative models rather than simply adopting the first and most obvious solution." Maier [21] defended that the evaluation of data modeling needs to be focused on criteria reflecting the quality of the modeling process and take into account different application scenarios of said modeling within an organization.

### 2.3.1 Definition of Criteria

William C. McGee [23], in his 1976 paper, devised a method for measuring a data model's quality, comprised of two main phases, firstly the identification of a set of model evaluation criteria (consisting in desirable attributes of a model), and secondly, the assignment of individual values to each item of the aforementioned criteria. Combining the weighted individual values of each criterion, one could easily calculate the model's suitability for a given data representation problem.

Accompanying this methodology, McGee defined twelve evaluation measurements, dividing them into two main groups: *use criteria*, measuring the model's ease of use, and *implementation criteria*¸ measuring the ease of the model's implementability as well as the efficiency of the resulting implementation.

Regarding *use criteria,* the following metrics were defined:

- **Simplicity**: A simple data model is described as having the "smallest possible number of structure types, composition rules and attributes";
- **Elegance**: A model is considered elegant if is capable of achieving a direct modeling capability with the least possible number of structures types, composition rules, and attributes;
- **Picturability**: All the structures defined in the data model should be able to be displayed in a graphic format;
- **Modeling directness**: The entities and relations presented in the model should have the maximum possible real-world counterparts;

- **Modeling uniqueness**: A model "should not provide equivalent direct modeling techniques", i.e., ideally, a model should provide only one way of modeling a certain situation that can be formulated with the terms defined in its knowledge base, in order to attain maximum performance;
- **Provision of schemas**: A model should contain structure schemas which allow data definition;
- **Implementation independence**: Data models usually contain convenient features for the implementers that are not necessary for the information modeling itself. Since these usually result in program dependence on them, they should be avoided;
- **Overlap with co-resident models**: A model should merge smoothly with other already-existing models;
- **Partitionability**: A model should contain structures that simplify the processes of partitioning data;
- **Non-conflicting terminology**: The terminology used by a data model should never conflict with already established terminology;

Regarding *implementation criteria*, two metrics were defined:

- **Proximity to the base model**: considered by the author as one of the most important model attribute from both an implementation and performance standpoint, a model should never be far from its implementation base, since whenever "structures of the model have exact counterparts in the base, it is unnecessary to develop procedures for transforming model structures into base structures and vice versa";
- **Applicability of safe implementation techniques**: A model should allow the use of already proved and well-understood implementation techniques.

Moody et al. [22], devised a comparison framework composed of five main constructs:

- **Qualities**: Defined as desirable properties or dimensions of value in data models, with the goal of the evaluation process being the maximization of the model regarding such properties;
- **Stakeholders**: The authors consider the contemplation of the unique needs of the different stakeholders who interact with the data models highly important. Therefore, in order to correctly classify a data model, one needs to have in mind the perspectives of

owners and customers, as well as the data analyst's, the application users', and the data administrators';

- **Metrics**: The authors define "consistent and objective" ways of classifying the quality of a data model. While there is the possibility that several different measures exist for the same quality, an overall measurement may be attained, through the combination of the ratings for all the metric relating to it;

- **Weighting**: Another important construct defined by the authors, is the attribution of weights to each of the model's qualities, representing said quality's importance on the scope of the project;

- **Strategies**: Finally, the authors conceptualize strategies for the improvement of the data model. Instead of focusing solely on the model's qualities and flaws, one should also focus on defining methods for improving the models.

In the same paper, to complement the aforementioned framework, the authors provide a set of qualities to have in mind for the evaluation of the data models:

- **Simplicity**: Regarding the aspects of size and complexity of the data model, namely the number of contracts required, the authors note that, as a general rule, the simpler models are usually the best ones, since they provide more flexibility, easier implementations, and are easier to understand. According to the authors, "One of the major goals of data modeling should be to search for simpler and more powerful ways of representing the data, rather than just documenting user requirements". Proposed metrics for this quality are the measurement of the data model complexity, measured by the number of entities plus the number of relationships in the data model;

- **Completeness**: Regarding the ability of the data model to "meet all user information and functional requirements", complete and correct requirements information is considered to be a crucial prerequisite for a successful development of information systems. Proposed metrics for this quality consist of ratings given by user and industry specialists, process mapping and package comparison;

- **Flexibility**: To be considered flexible, a data model should be easily adaptable to changes in the requirements. This quality is rather important since the "ability of a system to adapt to changes in its environment is widely considered as one of its most important characteristics". Metrics for the evaluation of this quality include senior management ratings, industry expert ratings, as well as data modeler ratings;

- **Integration**: To be defined as easily integrable, there needs to be a consistency of the information defined in the data model with the rest of the organization's data. Metrics for this quality consistent in a characterization of corporate data model conflicts;

- **Understandability**: this concept can be characterized as the "ease with which the concepts and structures in the data model can be understood by the users of the model". Evaluation metrics for this characteristic consist in ratings provided by users, data administrators, and application developers;

- **Implementability**: The implementability aspect of a data model can be defined as the ease with which the data model can be implemented, regarding aspects as time, budget, resource and technology constraints of the project.

### 2.3.2 Quantification of Qualitative Criteria

One of the most widely used methods of quantification of qualitative measurements is the employment of a Likert Scale. This psychometric tool was first devised in 1932 by psychologist Rensis Likert as a means to "measure 'attitude' in a scientifically and validated manner" [24], where attitude can be defined as a preferential way of "behaving/reacting in a specific circumstance rooted in a relatively enduring organization of belief and ideas" [24].

The Likert scale consists of a series of four or more questions (Likert items), each having a set of possible responses (originally five options), usually ranging from strong approval to strong disapproval of the concept. These responses are then combined in order to create an attitudinal measurement scale [25], objectively reflecting the test subject's opinion on a certain phenomenon.

Although five is the most common number of answers provided for a Likert item, some variations are also widely used. One example of said variations is the adoption of an even (asymmetric) scale, in order to remove the neutral answer (middle of the scale). This variation is often used when the researcher wishes to avoid the *central tendency bias*, a phenomenon that occurs when the survey participants wish to avoid expressing an opinion or because of *respondent fatigue* (which occurs when survey participants become tired of the survey task) [26]. Another frequent variant of the five point scale is the use of a seven or nine point scale. These types of scales are used when the researcher wishes to offer a bigger spectrum of options to his test audience, thus offering the participant the possibility of choosing a more precise answer, resulting in a greater probability of fitting the subject's perspectives.

Joshi et al [24] identified three principles for the correct construction of a Likert scale

1. The minimum score one can secure for the first three items of a scale is 3

2. Each numeral (response option) conveys the same meaning in all of the survey's items

3. All the scale items can be clubbed while satisfying the content and criterion validity

In its majority, due to the restrictions implied by the third principle, a new concept was created, in order to apply the same principles of the Likert Scale to a set of non-related questions. This kind of scale is called a Likert-type scale, and is used in situations where there is no attempt by the researcher to combine the responses from the items into a composite scale, but rather when the primary objective of the researcher is to study the opinions, beliefs or feelings a certain phenomenon causes in the test audience, instead of the generalization of the stance of the participants.

In their 2015 paper, Joshi et al. [24] also differentiate the process of data analysis based on the type of scale used. When dealing with Likert-type scale data, the values assigned to the items express a *greater than* relationship, with the size of the comparison being undefined. Given the uncertainty of this measurements, the authors consider this type of data to fall under the ordinal measurement scale. Due to this fact, the authors recommend the use of descriptive statistic tools, such as "modes or median values for central tendency and frequencies for variability".

On the other hand, when dealing with Likert scale data, the items are created by calculating a composite score from four or more items through the use of a sum or a mean, the authors state that the composite score should be analyzed at the interval measurement scale. For the interpretation and analysis of the results, the authors recommend the use of descriptive statistical tools, such as "the mean for central tendency and standard deviations for variability".

## 2.4 Conclusions

Great efforts are currently being made in the fields of data interoperability within IoT, and more precisely smart city systems.

The concept model developed by the British Standards Institute, mentioned in Section 2.2.2, provides a good starting point for the task in hands since the achievement of syntactic interoperability is the basis for the achievement of semantic interoperability. However, and according to the authors, "Sharing data across a city requires more than the interoperability covered by the SCCM." [17]. The LOV4IoT solution, presented by Gyrard et al. [18], also offers some advancements in the fields of data interoperability, by being able to aggregate ontology based projects among smart city sectors and finding common grounds between them. However, the platform is currently unavailable, thus providing no contributions for this thesis.

The solution for the problem previously mentioned will take into account the research and study of currently available data models for the IoT, and the construction of a comparison framework in order to categorize them. While the works developed by Moody [22], Maier [21] and, most importantly, by McGee [23] provide a good stepping stone for the evaluation of data models, they can be considered dated and not focused on the intricacies of the problems presented

by IoT systems and subsequent communication protocols. In order to cover these unaddressed
2 dimensions, additional criteria will be devised for the development of the framework.

# Chapter 3

# Data Models: Exploring a Solution for Citibrain

This chapter contains an overview of the problem to be addressed in this dissertation, and exposes the developed solution, detailing its methodology, architecture and the definition of evaluation criteria for the data models.

## 3.1 Problem Overview

Smart city data is obtained through sensory devices that measure different types of observations, such as light, movement or temperature. These sensors provide data of different and changing quality, and, with this data often being continuous, the feed will result in different validity and availability over time, translating to highly dynamic data streams [4].

Smart city data is also vast and heterogeneous in nature: being obtained from many different sectors (e.g.: traffic information, home automation, environment quality); being of different nature even within the same sector, i.e. to measure environment quality one would have to access metrics such as humidity, temperature and gas concentration levels in the atmosphere; as well as having different update rates.

Bischof et al. [4] identify three different types of update rate related to the data obtained from a city:

- **Static Data**: Data never or rarely updated (e.g.: points of interest);
- **Semi-Dynamic Data**: Data updated periodically (e.g.: cultural events);
- **Dynamic Data**: Data continuously updated (e.g.: traffic flow).

**Table 1: Information potentially retrieved from cities** [4]

| Data Category | Owner (Data Publisher) | Data Description | Sampling |
|---|---|---|---|
| **Transport** | Traffic Authority | Maps of Cities (Roads, Street Names, POIs, etc) | Static |
| | Municipality | Public Transport Schedules | Semi-Dynamic |
| | Traffic Authority | Transport Authority Updates (Roadwork, traffic status, etc) | Dynamic |
| **Air Quality** | Environment Agency | Particle Concentration | Dynamic |
| **Traffic** | Traffic Authority | Number of vehicles passing between two points, speed | Dynamic |
| **City Events** | Cultural Groups | Entertainment (movie/theater plays) | Semi-Dynamic |
| **Municipal Services** | Municipality | Library Data | Dynamic |
| | Private Company | Parking Meters | Dynamic |
| **Citizen Data** | Private Individuals | Social Media Information: Tweets, Status updates and blog posts, popular places ("check-ins") | Semi-Dynamic |
| | | Household Energy Consumption | Semi-Dynamic |
| **Health Data** | Private and Public | Relevant information about potential or confirmed sources of health threats | Dynamic |

After being collected, this data is usually transferred over several systems, being integrated into several lower-level (as is the case for complex sensors) or high-level (end-user) applications, as well as being made available via query or publish/subscribe services [4]. All these processes create additional levels of heterogeneity. For a better understanding of this smart city data flux, the following example will illustrate a concrete use case for the measurement of air quality in a city zone. An air quality sensor is usually a complex sensor, composed of several smaller sensors, responsible for measuring metrics such as temperature, gas concentration, luminance, precipitation, and noise levels. The complex air quality sensor is then responsible for the aggregation of the information provided by the small sensors, and sending it to an air quality management system, which will analyze the data and adapt it into the platform's data models. Besides the analysis by the management system, the processed, structured information is usually made available by the platform to websites, mobile applications and in the aforementioned publish/subscribe services, in the form of an API server. All these operations manipulate the sensed data and add different types of metadata (headers, representational terms) in order to better adapt the information to the systems, but, at the same time, creating barriers to the interoperability between the system and outer systems.

Extracting meaningful information from a smart city is a somewhat complex task, given the co-existence of the aforementioned levels of heterogeneity created by the information flux and the overall volume of the obtained data. Then, research on the processes of data interoperability becomes of primary interest [10].

As a result of the vastness and diversity of data, combined with the ever-increasing size of the IoT market, the heterogeneity problem goes even further. As the IoT, and specifically the smart city market continues to grow and get increasingly more attention by both the industry and the research community, IoT-powered applications are being rapidly developed in a great number of domains, such as environment, mobility, and healthcare. In order to keep track with this growth, companies and organizations have been betting in the development of new applications for the IoT, often resulting in the emergence of heterogeneous architectures, standards, middleware platforms and applications [27].

Kazmi et al. [27] identified three areas of the IoT where heterogeneity is introduced through the aforesaid developments, and highlighted some efforts currently being made by organizations:

- **Architectures and Standards**: Organizations and consortiums such as Fiware[4], Open Mobile Alliance[5], Eclipse IoT[6], the World Wide Web Consortium (W3C)[7], IPSO Alliance[8], the European Telecommunications Standards Institute (ETSI)[9], and the Internet Engineering Task Force (IETF)[10] have been developing new standards towards the harmonization of device communication.

- **Middleware Platforms**: In order to ease the collection of data from homogeneous and heterogeneous sources, a number of middleware platforms have been developed, such as Hayo[11], Cisco Flare[12], and SensorWare[13]. These platforms provide a plethora of functionalities, such as the collection of data from physical sensors, the transformation of such data into new structures and the enablement of interfaces for applications, and are usually also combined with other ICT technologies such as cloud computing. One of the bigger goals of these platforms is the encouragement of end-users to connect their IoT devices to the cloud infrastructure and the enablement of APIs for the attainment of sensory information and the development of applications.

- **Applications**: Recent times have witnessed the development of many IoT applications, either for research purposes or for commercial ones. These applications are often developed for very specific domains of a smart city, such as smart parking or waste management solutions, and very frequently make use of numerous IoT devices.

[4] www.fiware.org
[5] openmobilealliance.org
[6] iot.eclipse.org
[7] www.w3.org
[8] www.ipso-alliance.org
[9] www.etsi.org
[10] www.ietf.org
[11] www.hayo.io
[12] developer.cisco.com/site/flare
[13] sensorware.sourceforge.net

## 3.2 Data Models and Sematic Interoperability

This section contains a description of the proposed solution developed during this project. Firstly, a description of Citibrain's current platform is made, characterizing the platform's vertical solutions and its database structure. Secondly, the used methodology is explained, describing with detail the four phases that compose the project development, as well as the adopted approach for the evaluation and selection of the most suitable data model for Citibrain. Finally, the solution's architecture is presented, unfolding the several layers that compose it, how it integrates Citibrain's current architecture, as well as the definition of the technological stack used in the development of the final product.

### 3.2.1 Requirements

As stated in previous chapters, Citibrain currently offers four vertical solutions in the fields of mobility and environment quality: Smart Environment Management, Smart Parking Management, Smart Traffic Management and Smart Waste Management.

Sharing the same stack and technological architecture, Citibrain's architecture defines three main data model entities for each vertical solution – *Area*, *Thing*, and *Event* - as well as a horizontal entity, *Asset*, which is present across all solutions.

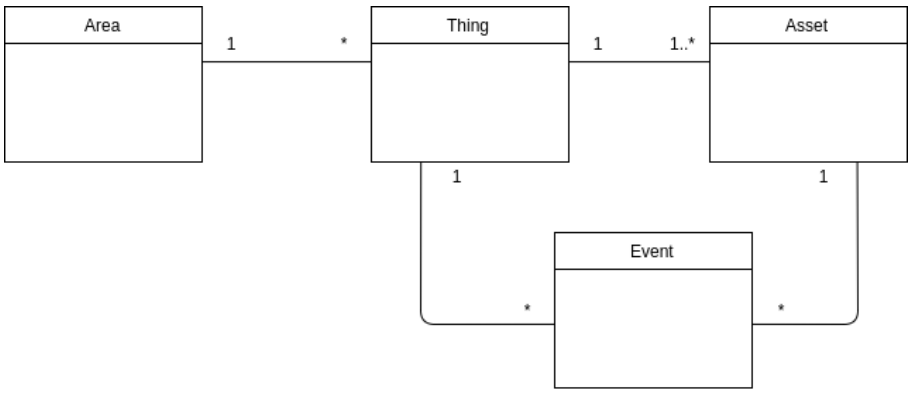



**Figure 5: Citibrain's generalized entity architecture**

An *Area* represents a geographically delimited space, delineated by a polygon, containing one or more *Things*, physical objects or places whose features of interest are the subject of the *Assets'* measurements. *Assets* represent Citibrain's sensory devices, which are responsible for the detection of changes in the *Thing*'s status, with those measurements resulting in *Events*.

Citibrain's *Assets* contain seven main attributes: (I) *uuid*, an Universally Unique Identifier, (ii) *type*, a variable of the type String which identifies if the device is either a sensor or an actuator, (iii) *name*, identifying the Asset by a sequence of characters, (iv) *is_active*, a boolean value indicating if the device is currently working at its full capacities, (v) *location*, a geofield of the

type *Point*, indicating the device's current geographical location by GPS coordinates, (vi) *battery*, an integer value representing the battery status of the device, and (vii) *status*, representing the device's possible states – ready, active, inactive or under repair. The details of the four vertical solutions' specific entities, as well as their relations are described in the next subsubsections.

Due to the system agnostic nature and overall platform independence of data models, Citibrain's requirements for new data structures consist only in the ability of the model to represent all solution's entities and relations. Other relevant characteristics of the solution comprise the minimization of overheads introduced by the models in the conversion, storage, and communication protocols of the current platform.

#### 3.2.1.1 Smart Air Quality

Citibrain's environment management solution has the purpose of monitoring a city zone's overall air quality. This vertical solution is composed of several small sensing stations, installed in the urban infrastructure, such as vertical signing, street lighting or electrical posts, which are capable of drawing indicators on air quality, noise pollution levels, temperature, atmospheric pressure, humidity and luminosity. These stations communicate in real time with the *Air Quality Management System* through Wi-Fi or GPRS, and are powered energetically by the structure they are installed in.

Regarding the solution's architecture, Citibrain's Smart Air Quality defines two main entities: *Assets*, representing sensors, and *EnvironmentEvents*, representing measurements performed by the sensors. An *EnvironmentEvent* is characterized by a *timestamp*, a *datetime* field representing the time and date of the measurement's occurrence, and 15 different measurements: (i) *temperature*, measured in degree Celsius, (ii) *precipitation*, measured in Impulses, (iii) *noise level*, measured in Decibels, (iv) *particles*, measured in percentage opacity, (v) *carbon dioxide*, measured in parts per million, (vi) *ozone*, measured in Volt, (vii) h*umidity*, measured in percentage, (viii*) wind speed* measured in Kilometer per hour, (ix) *light*, measured in Lux, (x) *solar radiation*, measured in watt per square meter, (xi) *carbon monoxide*, (xii) *nitrogen dioxide*, (xii) *volatile organic compounds*, (xiii) *processed ozone*, and (xv) *processed nitrogen dioxide*, all measured in Volt.

These entities and their relations can be observed in the figure bellow.

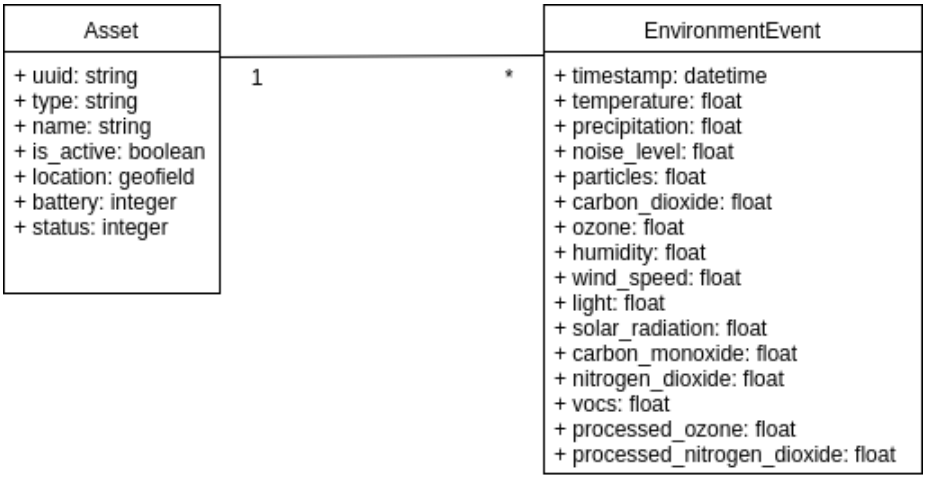



**Figure 6: Citibrain's Smart Air Quality solution's entity architecture**

#### 3.2.1.2 Smart Parking

Citibrain's Smart Parking solution has the objective of optimizing the searching task of available parking spots to its users, diminishing the emission of pollutants into the atmosphere. This solution is composed by six main features, (i) sensor-based vehicle detection, providing information about parking place availability, which is sent in real time to the parking management system, (ii) a communication gateway, (iii) digital parking panels, for informing the drivers about the number of available parking spots in certain zones, (iv) payment options, such as kiosks, mobile applications and SMS, (v) a mobile application for drivers, indicating available places and mapping routes to them, and (vi) an parking management system, responsible for the management of all parking related data.

Besides *Assets*, which represent magnetic sensors capable of wireless communication through various protocols, Citibrain's Smart Parking Management defines three entities, *ParkingArea, ParkingSpot* and *ParkingEvent. ParkingArea* represents a street zone with parking spots, and contains three fields, the *name* of the zone, the *default rate* in euros per hour, and *location,* a geographical polygon composed of three or more GPS coordinates. *Parking Spots* represent geographical *Points* in space where parking is allowed, and *ParkingEvent*s represent periods of time where a car was parked in a *ParkingSpot*. These events are characterized by three fields, *start timestamp,* a datetime field indicating the moment a car parked in the spot, *end timestamp*, a datetime field indicating the moment a car left the parking spot, and *is infraction*, a boolean field indicating if the car is parked illegally in the spot. The described entities and their relations can be observed in the figure below.

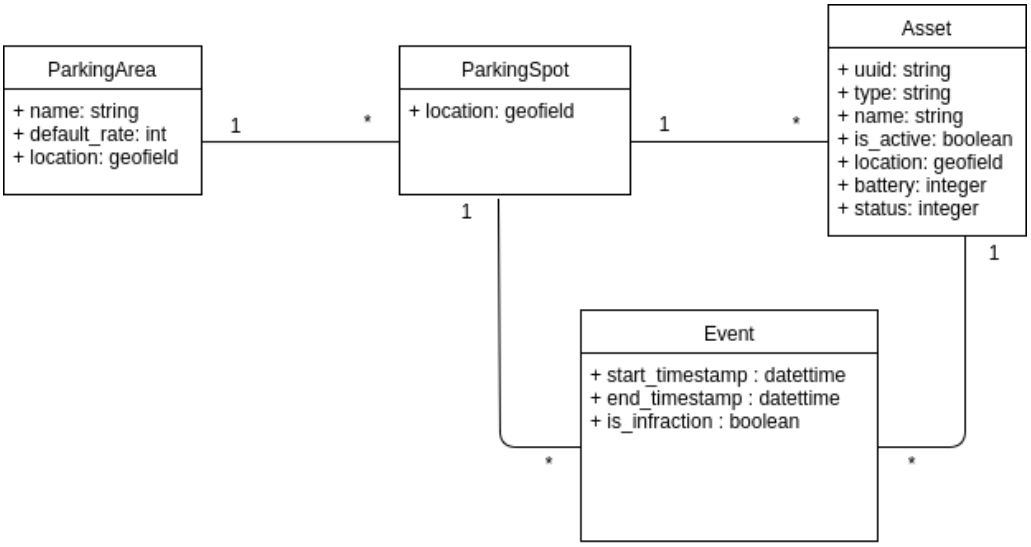



**Figure 7: Citibrain's Smart Parking solution architecture**

#### 3.2.1.3 Smart Traffic

Citibrain's Smart Traffic Solution has the goal of evaluating the current traffic flow in a city's roads, in order to understand the citizen's movement within a city's premises. The vertical solution is composed of two main features: (i) sensory devices, responsible for transmitting the volume and flow of traffic through gateways, and (ii) a central system, responsible for inferring inputs from the sensor networks and detecting and predicting traffic congestion.

This vertical solution contains two main entities, *assets*, corresponding to installed sensors throughout the city, and *traffic events* representing measurements provided by the sensors. These events are characterized by two fields, a *timestamp*, indicating the time and date the observation was performed, and *count*, an integer indicating the volume of cars counted since the last measurement.


**Figure 8: Citibrain's Smart Traffic solution architecture**

#### 3.2.1.4 Smart Waste

Citibrain's Smart Waste Solution has the objective of analyzing the filling level of a city's waste containers, in order to optimize garbage collection. The vertical solution contains three main characteristics, (i) the efficient management of waste collection routes, (ii) real time monitoring of container status via ultrasound or infrared measurements, and (iii) temperature monitoring of the container or box

This solution is composed by two main actors: *collection points*, which consist of containers with sensors responsible for informing about their occupational state and overall well-being, and the *waste management system*, responsible for processing the collection point's information and sending the most efficient routes to the collection trucks, with the goal of decreasing fuel costs.

Regarding the database structure, besides *Assets*, the solution defines three main entities, *WasteArea*, representing a geographical area containing several waste containers, and characterized by a *name*, a geographical polygon defined by three or more GPS points, and a *fill level alert*. The *WasteContainers* are defined by their shape, height and capacity, and contain information relating their geographical location (GPS point) and the date and time of the last waste collection (*last_collection*). The measurements performed by the sensors inside the containers are denominated *WasteEvents*, and are characterized by a timestamp of the observation, the container's fill level, temperature and information about their status (*is_turned_over*), indicating if the container has been tampered with. The described entities and their relations can be consulted in the figure below.

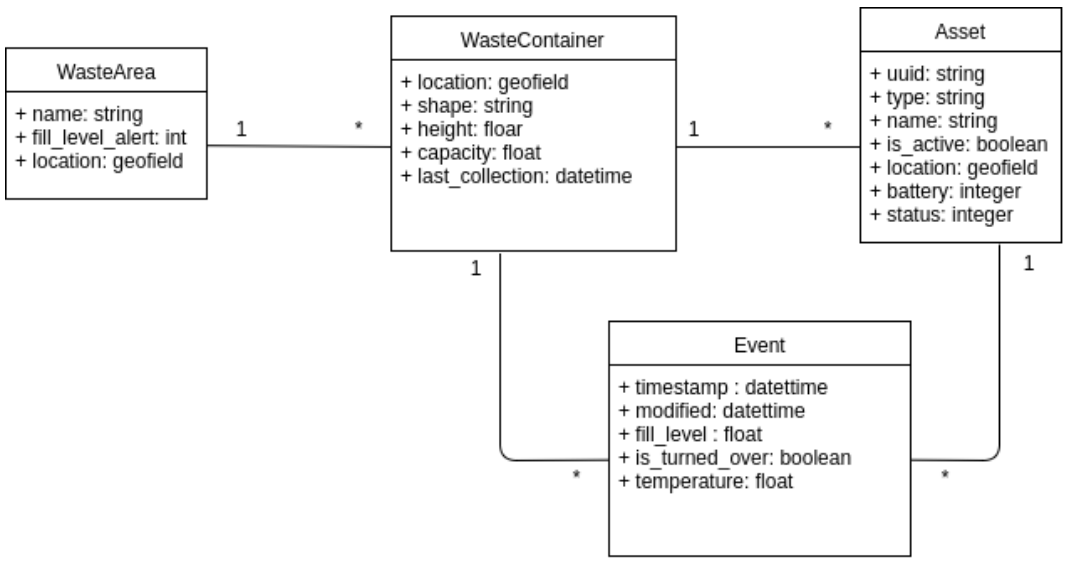



**Figure 9: Citibrain's Smart Waste solution architecture**

### 3.2.2 Methodology

The project was developed in five main phases: (i) a preliminary characterization of the discovered data models, culminating in the selection of the ones that fulfilled all defined requirements, (ii) the implementation of the selected data models to all four Citibrain's vertical solutions' data, (iii) the definition of the data models' evaluation criteria and construction of a formula capable of translating the evaluation of the data models into a single value for easy comparison, (iv) a further analysis of the implemented data models, ending in the selection of the most suitable data model for Citibrain's platform, according to the previously defined requirements, and (v) the development of a dashboard application, using the selected data model, serving as project validation.

The first phase consisted in researching for available data models for the IoT, and performing a characterization regarding their abstraction level. This characterization involved determining if the models are either abstract or instantiated, and, if the case is the latter, if (i) the available models cover all Citibrain's verticals, and (ii) if they are expandable, i.e., if means are provided for the instantiation of additional data models. The purpose of this analysis was to determine which data models are able to display all of Citibrain's information, making them a viable solution for the platform.

The second phase consisted in the development of a semantic translator software application, responsible for querying Citibrain's database, converting the provided information into the selected data models' format, and making it available through API services.

The third phase consisted in the development of a data model evaluation framework, based on the studies described in Chapter 2, as well as additional metrics relevant for IoT systems performance.

The fourth phase consisted in a detailed analysis of each selected model, evaluating them in both qualitative and quantitative dimensions. The qualitative analysis consisted in an evaluation of the models using the criteria defined in Chapter 5, which was based on the study presented in section 3 of Chapter 2. The quantitative analysis consisted in an evaluation of the overheads introduced in both the server and in the communication by using the models, measured by implementation and runtime performance metrics to be described in section 5.1. Combining the results of the two analysis into a single value, through the application of a weighted formula, devised in 5.2, the model with the best performance was chosen as the *de facto* best fit for Citibrain's verticals, and was used in the development of the fifth and final phase.

The fifth phase consisted in the development of a dashboard application, functioning as a proof of concept and project validation. This application uses the selected data model for the information structure and server communication, and is capable of displaying to the user real information obtained from Citibrain's sensors.

The entire process of development of this dissertation followed Ubiwhere's scrum-based agile development methodology, consisting of two-week sprints, each involving development,

test and documentation phases. This methodology is also complemented with a weekly activity planning.

As project management technologies, the following tools and frameworks were used: (i) Redmine for project management and issue tracking, (ii) Ubiwhere Information System (UIS), as a Redmine complement, for the registry of daily efforts, translated into work hours, and (iii) git for project code version control.

### 3.2.3 Solution Architecture

The developed solution consists in a client-server dashboard application, with the server and the API server being placed as an additional layer on Citibrain's information broker, as can be observed in the figure below.

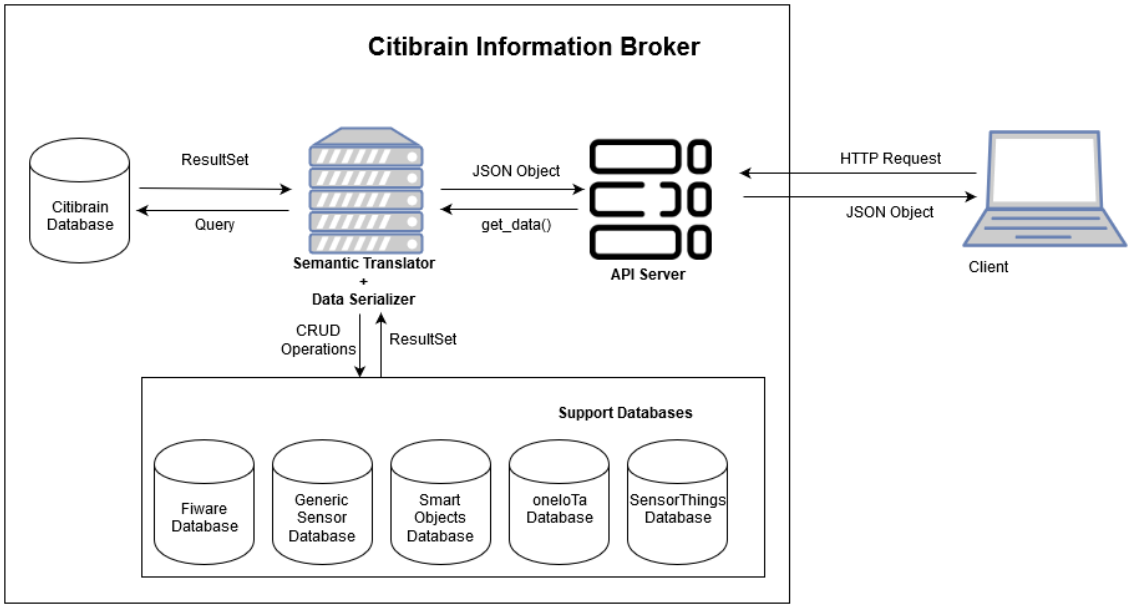



**Figure 10: Solution's Architecture**

#### 3.2.3.1 Database

For the development of the project, five new databases were created, each one corresponding to the implemented data model, whose architecture, entities and relations are comprehensively described in Chapter 6.

The creation of these additional databases also provides Citibrain the possibility of being compliant with five widely used IoT standards.

#### 3.2.3.2 Semantic Translator

The semantic translator is an additional layer placed within Citibrain's information broker, having two main purposes, (i) to convert Citibrain's information into the selected data model's structure, and (ii) to serialize Citibrain's data in the new format and make it available through an API. The data conversion is performed upon the submission of a command, where the semantic translator queries Citibrain's databases, converts the entirety of the original data into the new model's structure, and stores the resulting information in the corresponding database.

The API server is responsible for returning the requested data to the client, with the API request requiring the output data model as a parameter.

#### 3.2.3.3 Entity Interaction

As can be noted in Figure 9, the developed system is composed by five principal entities: (i) the semantic translator, (ii) the original Citibrain's database, (iii) the created support databases, (iv) the API server, and (v) the client. The main interactions in the system can be divided into two main stages, data conversion and data retrieval.

The data conversion stage represents the process of converting the original Citibrain sensorial data into the new data models by the semantic translator. This process is started by the issue of a command by the system administrator to the semantic translator. The translator will then query the bulk of the data contained in the original Citibrain database, and convert the information into the desired data model. The result is then stored in the corresponding database. A sequence diagram describing this interaction can be observed in the figure below.

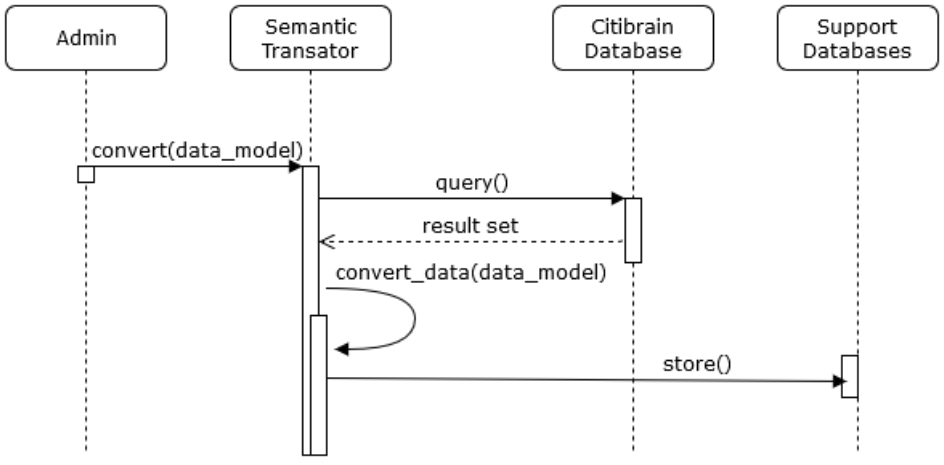



**Figure 11: Entity Interaction for Data Conversion**

The data retrieval stage represents the interaction between the system and the client. This process is started by the client issuing a HTTP request to the API server, having as parameters the

desired information and the data model to be used in the response. The API server interprets the request and sends a request of its own to the semantic translator. The translator is then responsible for querying the correct database, and serialize the result set into a JSON object, which is then passed onto the API server. Finally, the API server sends a response to the client with the desired information. A graphical representation of these interactions can be viewed in the following figure.

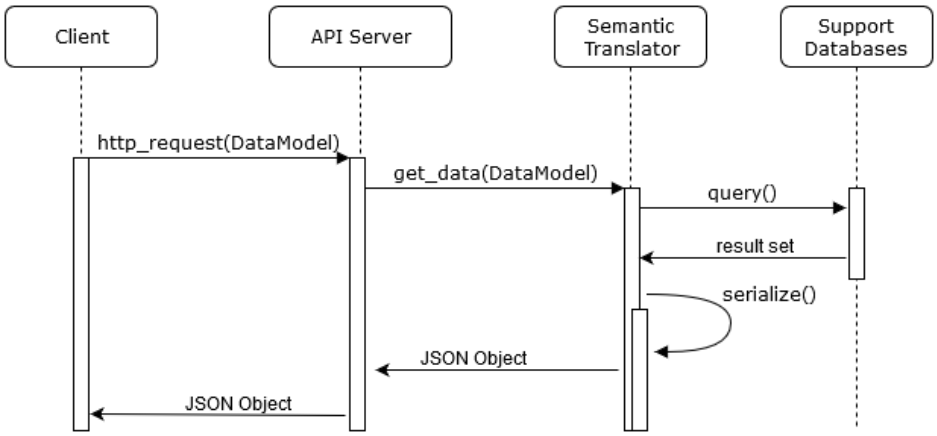



**Figure 12: Entity Interaction for Data Retrieval**

### 3.2.4 Proof-of-Concept Client

The client side of the project is a dashboard web application responsible for displaying sensorial information collected by Citibrain's sensors to its users, which is conveyed by API calls to the Server.

Regarding user interface, the application is composed of six main interfaces. When the user opens the app, they will be confronted with the welcome menu, where the chosen information can be consulted, regarding each vertical solution Citibrain disposes.

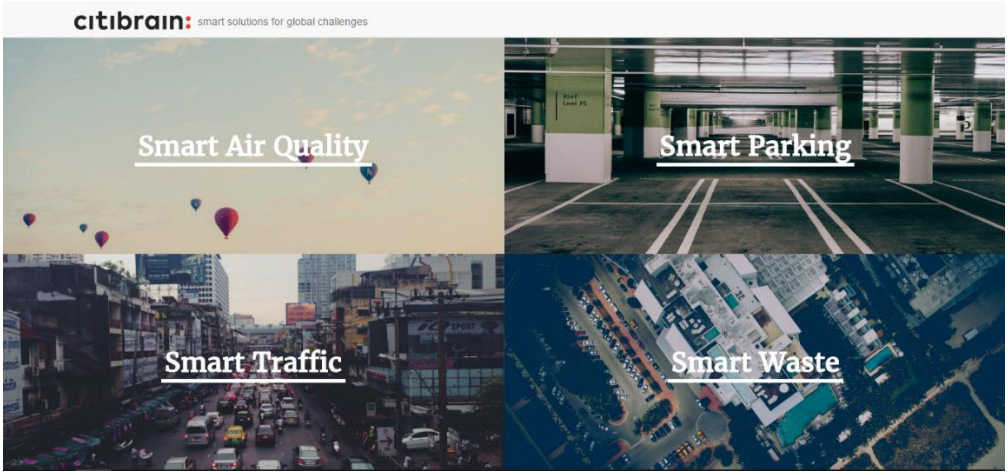



**Figure 13: Solution's main interface**

After selecting the desired vertical solution, the user will be redirected to the corresponding interface. All these interfaces share a similar structure in order to ease the dashboard's navigation, being composed by two main parts. The left side of the screen shows the user the location and area of the device they are consulting, while on the right side of the screen more specific information about the entity is shown, and information about the device's measurements can be consulted. The user also has the possibility of changing the geographical area being consulted, by clicking any of the presented arrows.

A comprehensive description of the other interfaces will be given in the next subsections.

#### 3.2.4.1 Smart Air Quality Management Interface

In this interface, the user is able to consult all the latest information regarding air quality collected by Citibrain's sensors. Besides temperature, precipitation, noise levels, humidity, wind speed, light and radiation on a selected area, the user is able to consult information regarding six air pollutant concentrations: (i) particles, (ii) carbon dioxide (CO2), (iii) ozone (O3), (iv) carbon monoxide (CO), (v) nitrogen dioxide (NO2), and (vi) volatile organic compounds (VOCs)


**Figure 14: Dashboard's Air Quality Management Interface**

### 3.2.4.2 Smart Parking Management Interface

The smart parking management interface gives the user information regarding specific street parking areas, such as the name and location of the area, information about the hourly rate, and the number of available spots. On the lower side of the right screen panel, the user is also able to consult when was the last update of the spot, its condition, and its location (by clicking the desired spot, a pinpoint will appear on the map).

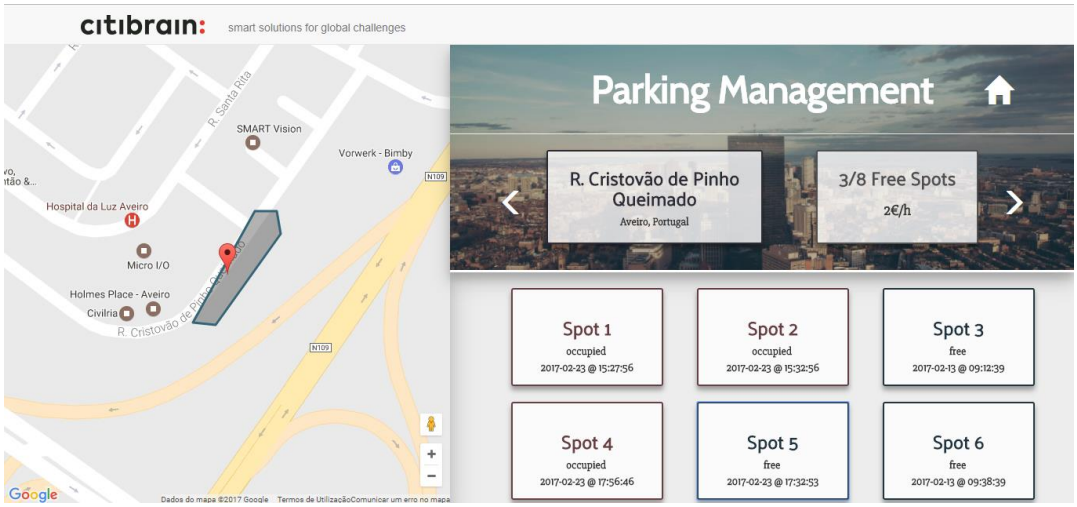



**Figure 15: Dashboard's Parking Management Interface**

#### 3.2.4.3 Smart Traffic Management Interface

Being one of Citibrain's simpler vertical solutions, the smart traffic management interface is only able to show the user information regarding the traffic flow of a city area. Emphasis is given to the last measurement (visible on the rightmost box of the upper panel of the interface), and a comparison with the last nine measurements is also made on the interface's bottom right panel.

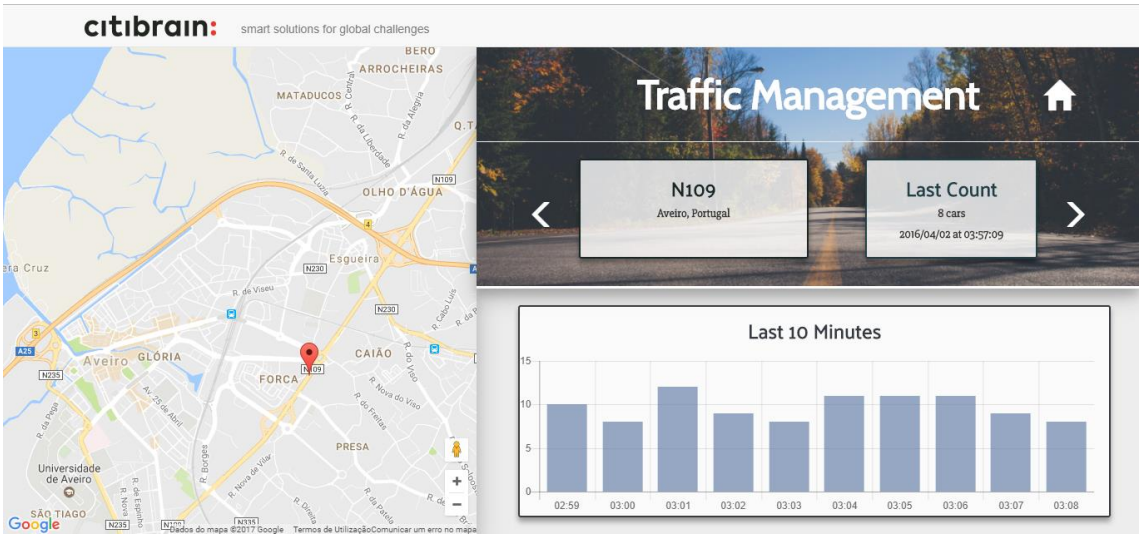



**Figure 16: Dashboard's Traffic Management Interface**

#### 3.2.4.4 Smart Waste Management Interface

The smart waste management interface is responsible for displaying the user information regarding the waste management areas in a city. By selecting one area, the same is delineated in the map present in the left panel, and information, such as the name of the area, the number of containers present in it and their state is shown on the right-side panel, as seen in the figure below.

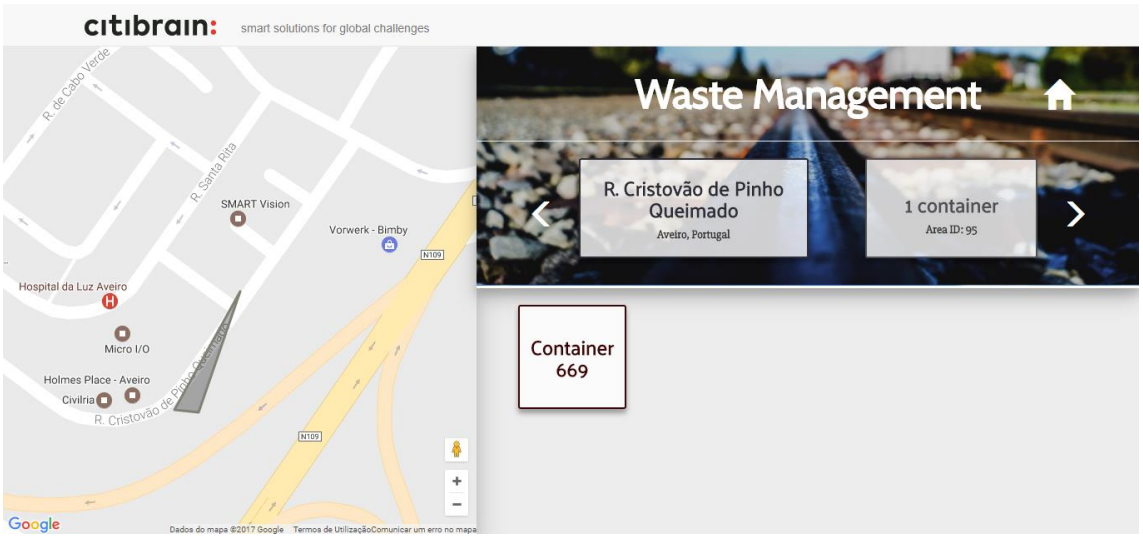



**Figure 17: Dashboard's Waste Management Interface**

By clicking any of the area's containers, the user is redirected to the Waste Container Information interface, as exemplified in Figure 18.

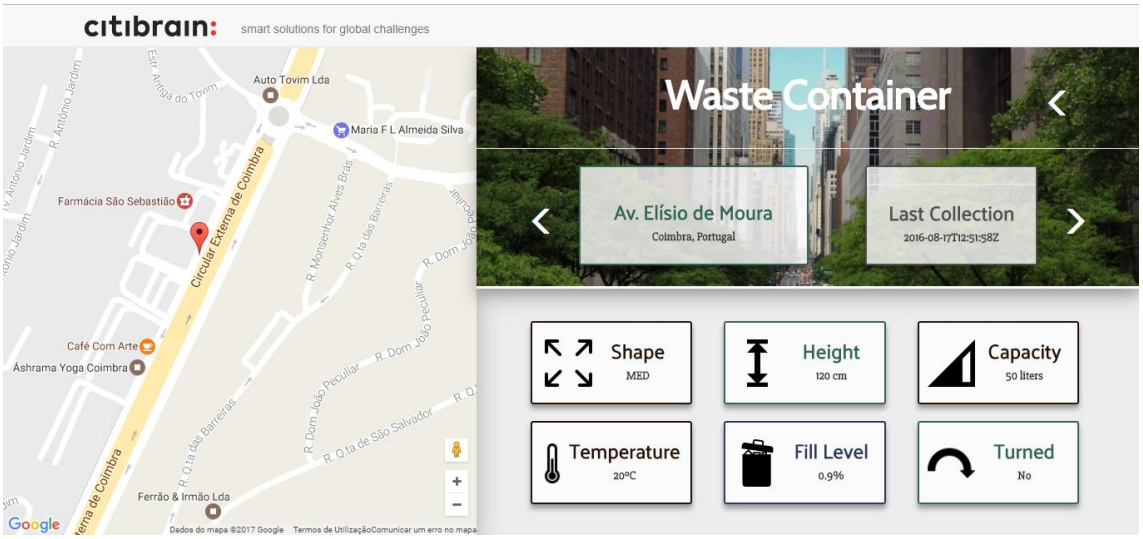



**Figure 18: Dashboard's Waste Container Interface**

Upon entering the waste container details interface, the user can consult information about the container itself – location, shape, height, capacity – and information about the current state of the container – temperature, fill level, turned over and last collection.

### 3.2.5 Implementation

The server was developed in Python[14] language, making use of its web framework Django[15], which follows the Model-View-Template architectural pattern, and Django Rest Framework[16] for data serialization and availability through API calls.

The conversion operation is achieved through a Python command with the generic form of:

python manage.py *model*-convert,

where *model* represents the desired data model for the conversion. After the submission of the command, the server queries Citibrain's databases, developed in PostgreSQL[17], converts the entirety of the original data into the new model's structure and stores the resulting information in the corresponding database, also developed in PostgreSQL.

For the implementation of the client dashboard application, Twitter Bootstrap[18] was used for the interface design, Angular 2[19] was used for the app's logic and server communication, and Google's *Material Design*[20] directives were followed in the development of the application's interfaces.

### 3.2.6 Selection of Data Model Evaluation Criteria

As stated in 3.2.2, during the project development, the data model evaluation process is composed by two phases.

Regarding the first phase, in the preliminary analysis, the models are characterized upon their abstraction level, where they can have one of two possible values: instantiated or abstract. While the instantiated models dispose of already pre-defined data structures for different metrics, such as air quality, acceleration or parking, the abstract models dispose only of the definition of the main entities in a single model, such as *sensors*, *observations* or *features*. If the models are instantiated, it is also important to determine two aspects: (i) their ability to represent Citibrain's entities and relations, and (ii) if they are expandable, i.e., it is possible to implement additional models.

Regarding the second phase, for the evaluation of the selected data models, the process is divided into two main analysis categories: (i) qualitative, referring to more non-measurable and subjective qualities of the data models, and (ii) quantitative, for a more precise quantification of

[14] www.python.org
[15] www.djangoproject.com
[16] www.django-rest-framework.org
[17] www.postgresql.org
[18] www.getbootstrap.com/
[19] www.angular.io/
[20] www.material.io/guidelines/

aspects such as computer resources spent in data conversion and the overheads introduced by the representation in both the representation and communication aspects.

# Chapter 4

# Data Models for the IoT

The goals of this chapter are twofold: firstly, a description of the seven discovered data models - FIWARE [28], OGC SensorThings [29], CitySDK[30]–[32], OData[33], W3C Generic Sensor [34], IPSO Smart Objects [35] and oneIoTa [36] – will be given, and secondly, a preliminary analysis of the models' suitability for Citibrain's vertical solutions will be performed.

For each model, the description will be as follows. In a first step, the abstraction level is identified. This property can have one of two possible values: (i) Instantiated, if a set of predefined data models for certain metrics and observations are provided, (ii) Abstract, if only one model is provided, which only defines the entities of the system and their relationships.

In a second step, if the models are instantiated, three dimensions are studied:

- The smart city sectors they apply to;
- The list of the available predefined models;
- The expandability of the models, i.e., if it is possible to develop non-provided models.

If the models are abstract, a description of the entities and their relations will be given.

Regarding the suitability analysis, the main goal will be to determine if the instantiated models dispose of structures capable of representing information related to the fields of air quality management, waste management, parking management and traffic management, which represent the four vertical solutions developed by Citibrain.

The descriptions of the models in the next sections are solely based on their provided specifications, except where noted.

## 4.1 CitySDK

CitySDK[21] is a project that characterizes itself as a "Service Development Kit", with the aim of harmonizing APIs across cities. The project is focused on three main smart city sectors: tourism, mobility and civic action, providing communication APIs and various instantiated data models for each of them.

Regarding *Mobility*, the project uses the Linked Data API, which offers "unified and direct access to open transport, mobility, and geodata" [32] from a city.

Regarding *Civic Issue Tracking*, the platform uses the Open311 API, which enables sending service requests to the city's feedback system [30]. The used data model for the API is the *Civic Issue Tracking* Data Model, which describes the type of service requested by the user, and its current resolved status.

Regarding the *Tourism* sector, the project offers its *Tourism* API, which focuses on creating location-based mobile services for tourists. Three data models are offered: *Points of Interest*, describing all the POIs available in a city, *Events*, describing a list of events either currently taking place or already concluded in the city, and *Routes*, providing routes for visiting the city's points of interest [31].

This platform has one big drawback, since it requires a bureaucratic process for the registration of the city in the project, in order to contribute to the platform, and implement and explore its functionalities. The toolkit also offers expandability, providing guidelines and enabling registered cities to contribute to the datasets and creating new functionalities based on the existing vocabularies and data models.

## 4.2 FIWARE

FIWARE[22] is a technological platform developed by the FIWARE Community, an independent and open community with the mission of building "an open sustainable ecosystem around public, royalty-free and implementation-driven software platform standards that will ease the development of new Smart Applications in multiple sectors" [28]. FIWARE provides an OpenStack-based cloud environment, complemented with a set of open standard APIs, with the goal of facilitating (i) the connection to the IoT, (ii) the processing and analysis of Big Data, and (iii) the incorporation of advanced features for user interaction [37].

The platform also defines its own RESTful over HTTP communication API, FIWARE NGSIv2 API which is based on Open Mobile Alliance's OMA NGSI API. Also making use of the Orion Context Broker for mediation between the consumer producers (such as sensors) and context consumer applications (such as smart phone applications) [38], FIWARE disposes of a

[21] www.citysdk.eu
[22] www.fiware.org

total of twelve instantiated data models in JSON format: (i) *Alarms*, (ii) *Parks and Gardens*, (iii) *Environment*, (iv) *Points of Interest*, (v) *Civic Issue Tracking*, (vi) *Street Lighting*, (vii) *Device*, (viii) *Transportation*, (ix) *Key Performance Indicators*, (x) *Waste Management*, (xi) *Parking*, and (xii) *Weather*.

The *Devices* data model is of particular interest since it can be applied to any type of sensor, actuator or any device that is capable of communicating electronically via a network, which provides a good starting point for expandability. This particular model is divided into two parts: *Device* for the storage of data produced by one of these tangible objects; and *DeviceModel*, for capturing static properties common to each instance of a *Device.*

## 4.3 W3C Generic Sensor

The Generic Sensor API[23] was developed by the World Wide Web Consortium[24] with the goal of promoting "consistency across sensor APIs, enable advanced use cases thanks to performant low-level APIs, and increase the pace at which new sensors can be exposed to the Web by simplifying the specification and implementation processes" [34].

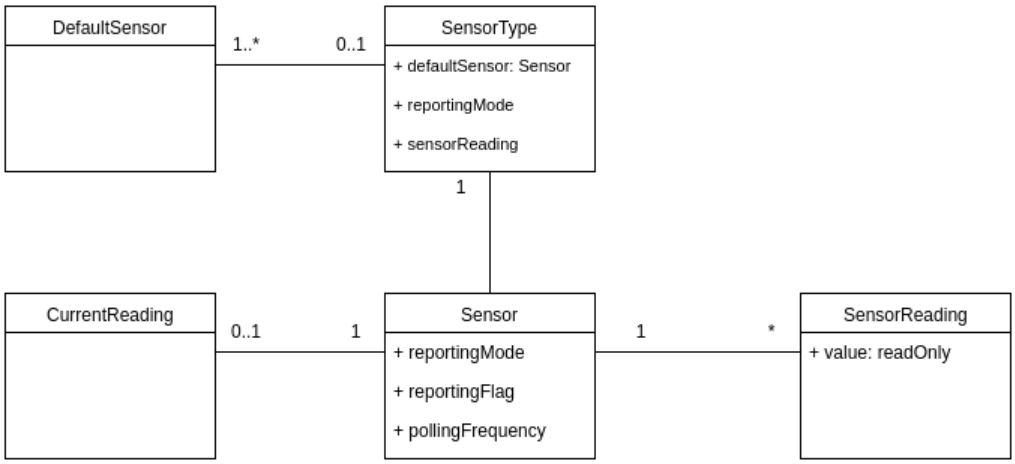



**Figure 19: Generic Sensor Architecture**

The API's data model specification focuses on the concepts of sensors and their readings.

A *sensor* provides a measure of a certain occurrence, defined as a *raw sensor reading*. Known to be somewhat discrepant to the actual physical value being measured by the sensor, these readings need to be corrected. If the discrepancy is predictable, the value is corrected by an operation provided by the API, known as *calibration.* Otherwise, the discrepancy needs to be addressed in a dynamic fashion by a process class, called *sensor fusion* (combination of various

[23] www.w3.org/TR/generic-sensor
[24] www.w3.org

sensor readings, carried either in hardware or software). The resulting values are referred to as *sensor readings.*

The specification also distinguishes sensors by their *sensor type*, being that different sensor types measure different physical characteristics (e.g.: water quality, temperature, air pressure). Two main groups are defined, based on a sensor's type, low-level sensors and high-level sensors, which usually result from the application of algorithms to low-level sensors.

Low-level sensors are characterized by their implementation, such as a Gyroscope, where high-level sensors are characterized by their readings, as is the case of geolocation. In geolocation, it is known that the provided information regards the user's location, however, the means of obtainment of the data are purposely left unspecified.

Lastly, the specification also defines the update rate of the sensors measurements, under the designation of *reporting modes*. If a sensor updates at a regular interval, an adjustable frequency in hertz, the *reporting mode* is considered *periodic*. Otherwise, if the update is made only when there is a change in the measurement value, the *reporting mode* is said to be *auto*.

The relations between the defined concepts are as follows:

- Regarding *Sensor Types*
  - *Sensor Types* have one or more associated *sensors*, an associated *Sensor* subclass, an associated *Sensor Reading* subclass, whose attributes must be read-only, a *default sensor*, a set of *supported reporting modes*, as well as an abstract operation to construct a *Sensor Reading* object. If a *Sensor Type* possesses two or more sensors, there is a need for it to have associated *identifying parameters*, which have the responsibility of selecting the correct *sensor* to be associated with each new *Sensor* object.
- Regarding *Sensors*
  - A *Sensor* is associated with a set of activated *Sensor Objects*, a current reading, a reporting flag, a current reporting mode, as well as a current polling frequency.

## 4.4 IPSO Smart Objects

Smart Objects is a protocol for machine to machine communication and IoT device management developed by IPSO Alliance (www.ipso-alliance.org). Based on Open Mobile Alliance's (OMA) Lightweight Specification, LWM2M, Smart Objects aims at providing a common design pattern (object model), to ensure high level interoperability between objects, devices and services. Aiming at a reusable design, IPSO Smart Objects doesn't rely on any specific communication protocol, being able to be used on top of any RESTful protocol. Smart

Objects main difference from LWM2M is that the defined objects do not have fixed mandatory resources.

Although defined as abstract, this data model is an instantiated object model, containing definitions for objects in XML format, available in two packs: (i) starter, containing the 18 most common objects, such as temperature sensors, light controllers, presence sensors, among others, and (ii) expansion, providing an additional 33 sensor, actuator and control switch definitions.

IPSO defines its data model as being divided into four parts, following the OMA LWM2M object model:

- **Object Representation**: objects are represented as a URI path with three 16-bit integers, in the form of *Object ID/Instance ID/Resource ID*. Objects define the semantic type of *Instances*, *Instances* represent specific object types at runtime, and *Resources* represent a view or active property of an object.

- **Data Types**: the model contains seven data types: (i) string, (ii) integer, (iii) float, (iv) boolean, (v) opaque (sequences of binary octets), (vi) time, and (vii) object link – to refer to an instance of a certain object.

- **Operations**: four operations are defined: (i) *Resource Values for Read*, *Write* and *Execute* operations, (ii) *Object Instances* for *Create* and *Delete* operations, *Objects* and their instances contain *Read* and *Write* operations, and *Attributes* have *Set* and *Discover Operations*.

- **Content Formats**: the model defines content formats for resources in the form of "text/plain" or "tlv", for object in the forms of "text/senML+json", "application/cbor", and "binary/tlv", and for attributes in the forms of "link-format" or "link-format+json".

Smart Object's object model architecture, also based on LWM2M's, contains three main entities: *Resources*, *Objects* and *Instances*, with the first two visible in Figure 20.

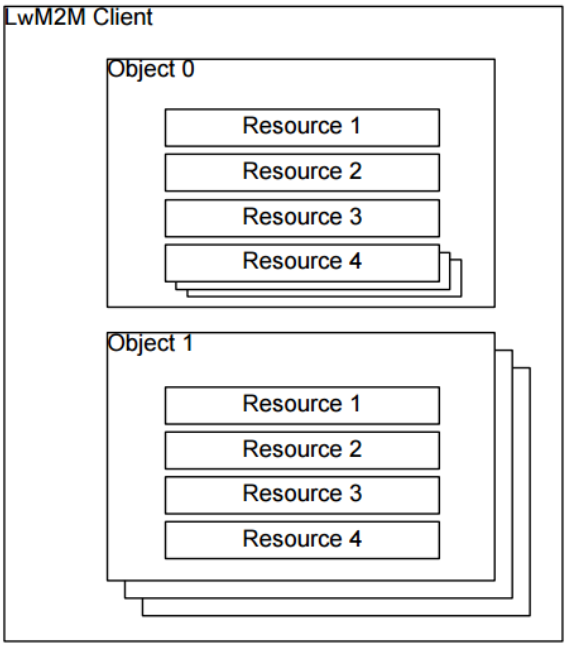



**Figure 20: IPSO/LWM2M's Object Model Architecture**

- **Resource:** Resources are **"**atomic pieces of information that can be Read, Written or Executed" [39];
- **Object:** Objects are "typed containers, which define the semantic type of instances." [35];
- **Instance**: Instances represent "specific object types at runtime, and allow Smart Object endpoints to expose multiple sensors and actuators of a particular type" [35].

To illustrate these definitions, the example of an object could be a generic sensor, with the sensor value, units, minimum measured units and maximum measured units as resources. In this case, instances of the object would be concrete sensors, placed in physical locations [37].

Both LWM2M and IPSO Smart Objects allow the creation of composite objects, i.e., objects composed of other objects. One example for this concept would be a complex sensor, capable of measuring temperature, time and humidity, as is the case of many air quality management sensors.

## 4.5 OData

OData[25] (Open Data Protocol) is a standard developed by OASIS[26] – Advancing Open Standards for the Information Society. It defines a set of best practices for the building and consumption of RESTful APIs, disposing of an abstract data model, the *Entity Data Model*.

The data model defines five central concepts:

- *Entities*, representing instances of *Entity Types*, which are "named structured types with a key" that "define the named properties and relationships of an entity" [31];
- *Relationships*, which constitute navigation properties;
- *Entity Sets*, which consist of named collections of entities;
- *Functions*, executions of custom logic without side effects, on sections of the data model;
- *Actions*, which represent executions of custom logic on parts of the data model with side effects.

The last two concepts are aggregated in the specification as *Operations.*

The relationships between two or more *Entity Types* are defined by *Associations*, which are grouped in *Association Sets*. *Entity Types* can also have *Navigation Properties*¸ which are destined to a specific association and can be used to refer to associations of an entity [40]

The model also provides the use of *Annotations* for the model and instance elements. *Annotations* provide the possibility of specifying individual facts about elements (e.g. if it is read-only or writable) or defining common concepts (e.g. a person).

OData also supports both JSON and AtomPub, an XML-based format for resource representation, and offers CRUD support, through four HTTP methods [40]

The OData service is compatible with any platform that provides support for XML and HTTP.

## 4.6 oneIoTa

oneIoTa[27] (one Internet of Things architecture) is a Data Model Tool developed by Open Connectivity Foundation[28], with the aim of "providing a common scalable standard for the

[25] www.odata.org
[26] www.oasis-open.org
[27] www.oneiota.org
[28] www.openconnectivity.org

Internet of Things with a certification tool that could ensure interoperability and an open-source implementation that could accelerate implementation time" [36].

This tool provides instantiated Data Models in both RESTful API Modeling Language (RAML) - for interface definition - and JavaScript Object Notation (JSON) - for schema definition - formats, obtained through crowdsourcing, mainly for the sectors of home automation and health.

Regarding the areas of mobility and environment quality, the tool only offers five models: *Air Quality*, *Temperature*, *Humidity*, *Geolocation* and *Energy Consumption*.

oneIoTa also provides an API designer console for expandability, on the basis that new data models are created from existing ones.

As a complement to this Data Model tool, OCF also offers IoTivity, an open-source software framework that provides machine-to-machine connectivity.

## 4.7 OGC SensorThings

OGC SensorThings[29] is a standard specification with the aim of providing an "open and unified way to interconnect IoT devices, data, and applications to the web" [29], developed by the Open Geospatial Consortium[30].

Part of the OGC Sensor Web Enablement standards, a global framework of standards and best practices, aiming for fast and practical linking of diverse sensor related technologies, this API provides an abstract data model, whose schematization is represented in Figure 21.

This Data Model consists of two main parts, sensing, and tasking, with the latter still in development. While the sensing profile, based on the ISO/OGC Observation and Measurement model, provides a standard way of managing and retrieving operations from heterogeneous IoT sensor systems, allowing applications to perform CRUD operations, the tasking profile will be responsible for the parametrization of IoT devices, such as sensors and actuators.

The API offers eight resources: (i) *Thing*, (ii) *Location*, (iii) *HistoricalLocations*, (iv) *Datastream*, (v) *ObservedProperty*, (vi) *Sensor*, (vii) *Observation*, and (viii) *FeaureOfInterest*, whose relationships are described next, according to the specification provided by OGC [41]:

- An *Observation*, classified by its event time and used procedure (e.g.: a sensor), is an act that produces a result, a value representing a *FeatureOfInterest*.
- A *Thing*, i.e. an object in the physical world, contains *Locations* and *Historical Locations* (the set of previous locations).

[29] www.github.com/opengeospatial/sensorthings
[30] www.opengeospatial.org

- A *Thing* also has *Datastreams*, collections of *Observations* – events performed by sensors that produce an estimate - that measure the same *ObservedProperty* and corresponding *Sensor*

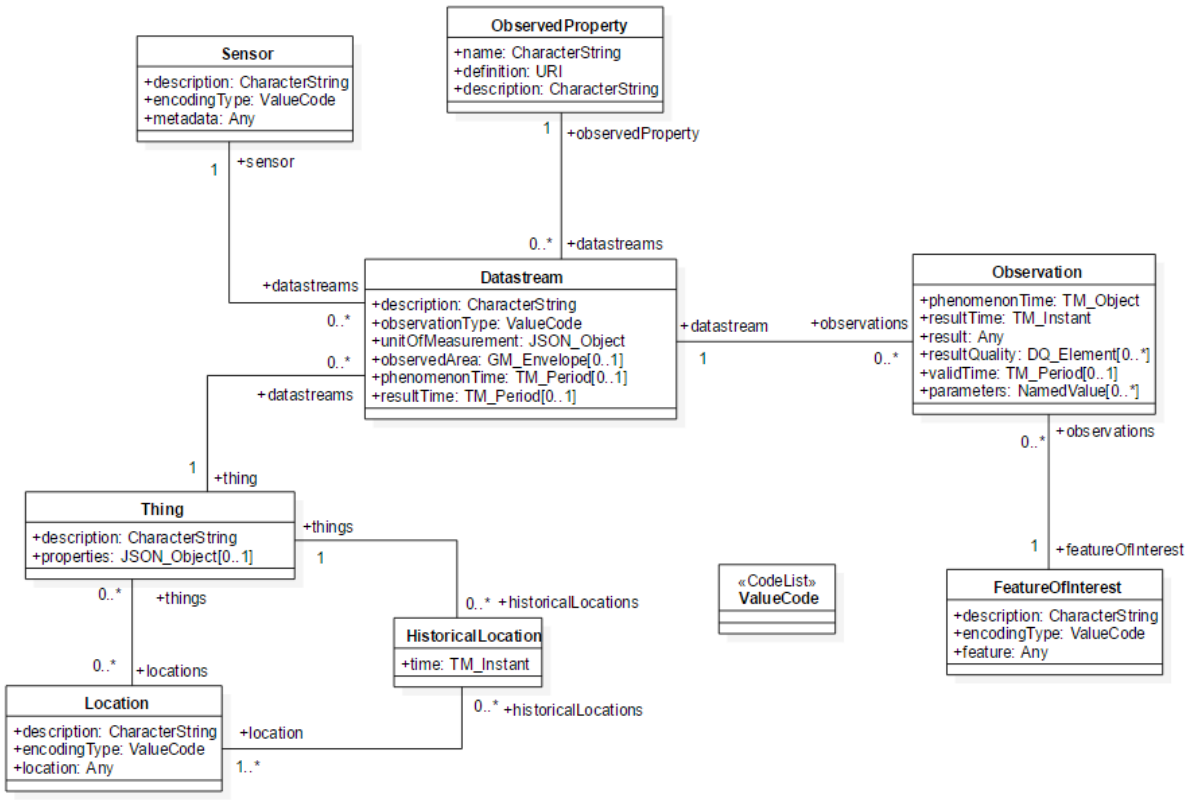



**Figure 21: SensorThings Sensing Profile Core Entities** [29]

## 4.8 Conclusions

After a preliminary study, some considerations can be made regarding the models' suitability for Citibrain's solutions, following the metrics described in the beginning of this chapter. The main goal of this preliminary characterization of the models is to determine if they are able to represent all of Citibrain's entities

In order to aid the comparison between the several models, the following table maps the entities of each model to the entities described in Citibrain's database.

**Table 2: Data Model Entity Mapping**

| Citibrain | Fiware | CitySDK | oneIoTa | SensorThings | OData | GenericSensor | Smart Objects |
|---|---|---|---|---|---|---|---|
| **Asset** | Device | N/A | Sensor | Sensor + Location + Datastream | Entity | Sensor + Sensor Type | Object - Sensor |
| **Environment Event** | AirQualityObserved | N/A | AirQuality + TimePeriod | Observation | Property | Reading | Object (Sensor Value + Units) |
| **Parking Area** | OnStreetParking | N/A | Area | --- | Entity Set | --- | Composite Object |
| **Parking Spot** | ParkingSpot | N/A | Parking Spot | Thing + Location + ObsProp + FOI | Entity | Included in Sensor and Type | Composite Object |
| **Park Event** | ParkingSpot (last event) | N/A | Occupancy + TimePeriod | Observation + ObsProp | Property | Reading | Object (Sensor Value + Units) |
| **Traffic Event** | TrafficFlowObserved | N/A | Activity Count + TimePeriod | Observation + ObsProp | Property | Reading | Object (Sensor Value + Units) |
| **Waste Area** | WasteContainerIsle | N/A | Area | --- | Entity Set | ---- | Composite Object |
| **Waste Container** | WasteContainer | N/A | Waste Container | Thing + ObsProp + FOI | Entity | Included in Sensor and Type | Composite Object |
| **Waste Event** | WasteContainer (last event) | N/A | Fill level + temperature + turned over + TimePeriod | Observation + ObsProp | Property | Reading | Object (Sensor Value + Units) |

- Fiware provides instantiated models for all Citibrain's sectors, is expandable and provides guidelines for the development of new data models. These data models are considered suitable for Citibrain.

- CitySDK provides instantiated models, fitting only one of Citibrain's sectors – mobility, and, even within that sector, provides data models for none of Citibrain's vertical solutions. Although expandable, there is a need for the registration of the city's municipality on the platform's website, which results in a possibly prolonged bureaucratic process. Due to the closed nature of the platform, it can be considered a bad solution for Citibrain.

- oneIoTa provides instantiated data models, fitting, however, only one of Citibrain's sectors – Air Quality management. Since the models are expandable, the platform can be considered a good solution for Citibrain, and data model contributions can be considered a good complement for the research and development aspects of this project.

- SensorThings, Generic Sensor, and Smart Objects are abstract models, meaning all four can be used in Citibrain's context.

- Since during the research process developed in this chapter little documentation was discovered for the OData data models, with the majority of it being oriented towards business management, it can be considered a bad fit for Citibrain.

**Table 3: Comparison of the data models' abstraction level**

| Name | API | Abstraction | Expandable | Mobility | Waste | Environment |
|---|---|---|---|---|---|---|
| **FIWARE** | NGSI v2 | Instantiated | Yes | Yes | Yes | Yes |
| **SensorThings** | SensorThings | Abstract | N/A | N/A | N/A | N/A |
| **CitySDK** | CitySDK | Instantiated | Yes | Yes | No | No |
| **OData** | Unspecified | Abstract | N/A | N/A | N/A | N/A |
| **GenericSensor** | GenericSensor | Abstract | N/A | N/A | N/A | N/A |
| **Smart Objects** | Unspecified | Abstract | N/A | N/A | N/A | N/A |
| **oneIoTa** | Iotivity | Instantiated | Yes | No | No | Yes |

Bearing in mind the aforementioned conclusions, five models have been selected for further study: FIWARE, SensorThings, Generic Sensor, Smart Objects, and oneIoTa.

# Chapter 5

# Criteria for Data Model Evaluation

This chapter contains the definition of the used criteria for the data model evaluation phase of the project. In the first section, a selection of criteria for the evaluation of the studied data model is made. In the following section, the chosen criteria are combined into a weighted sum, capable of translating a data model's characterization into a single value, thus allowing a consistent evaluation of each model. In the closing section, some considerations regarding the selected metrics and dimensions, and subsequent evaluation are made, mainly exposing the choice of the weights for the formula.

## 5.1 Formulation of Evaluation Criteria

This section presents the chosen criteria to be used in the data model's comprehensive analysis. This evaluation is to be divided into two main components: qualitative and quantitative. The qualitative metrics have the purpose of quantifying the less palpable characteristics of the data models, and are derived from the research carried out in Chapter 2. The quantitative metrics correspond to the more easily measurable qualities of the data models, related, in their majority to aspects of latency and communication loads.

### 5.1.1 Qualitative Evaluation

For the qualitative analysis, based on the metrics studied in section 2.3, a set of nine criteria was chosen. Of these criteria, seven metrics were selected from the analyzed publications: (i) overlap with co-resident models, (ii) simplicity, (iii) completeness, (iv) implementability, (v)

flexibility, (vi) understandability, and (vii) easiness of development. These criteria were deemed the most relevant of all the studied metrics, and representative of the desirable qualities of a data model. However, in order to better complement the analysis, two additional metrics were added to the qualitative analysis: (i) amount and quality of available documentation, and (ii) the existence of implementation tools, in order to achieve a more complete evaluation.

The definition and quantification process of the selected metrics are as follows:

- **Integration/Overlap with co-resident models**: The model's overlap with co-resident models is calculated as the consistency between its information structure and Citibrain's database's. In order to provide a more illustrative value, the metric is to be calculated as the amount of Citibrain's base model entities the new data model is able to represent;

- **Simplicity**: In the comparison process, the data model's simplicity is calculated by the sum of the number of described entities plus the number of relations present in the model;

- **Completeness**: For this evaluation, completeness is calculated as a ratio between the simplicities of the data model and Citibrain's base model;

- **Implementability:** The model's implementability is calculated by the number of hours spent in its implementation;

- **Flexibility**: The model's flexibility is calculated as a subjective analysis of its ability to be compliant with changes in the original database information structure. This metric is calculated through the use of a Likert-type survey;

- **Understandability**: The model's understandability is calculated as a subjective analysis of the ease with which a user with reasonable knowledge in data modeling and willingness to study the information will have to understand the concepts, structures and relations present in the model;

- **Easiness of development**: The model's easiness of development is calculated as a subjective analysis of the implementation experience, in the form of a grading system, more specifically, thought the user of a Likert-type survey;

- **Amount and quality of available documentation**: Regarding documentation, a quantitative study was performed on each data model, based on the following seven metrics: (i) Does the platform provide a specification for its data model? (ii) Do implementation guidelines exist? (iii) Are there any tutorials provided? (iv) Does a repository exist? (v) Are examples given of the models' implementation? (vi) Does a

community exist? If so, is it responsive? (vii) Are there any questions related to the models on *Stack Overflow?* If this is the case, do they have accepted answers?

- **Implementation Tools:** This metric will be calculated by the model's platform ability to answer the following questions: (i) Does the platform provide any kind of SDKs? (ii) Are there any support libraries available? (iii) Do any validation tools exist?

For the quantification of the last two metrics, the model is to be awarded one point for each question it can answer.

### 5.1.2 Quantitative Evaluation

Regarding the quantitative analysis, four criteria representative of desirable qualities in a data model for the IoT were selected:

- **Temporal Overhead**: The model's calculated temporal overhead is determined as the amount of time spent on the conversion from the original database information structure to the one present in the data model;
- **Memory Overhead**: The model's calculated memory overhead is determined as the increase in the computer's memory resources noted during the information conversion;
- **Representational Overhead**: The model's calculated representational overhead consists of the size, in bytes, of both the representation of a dataset and the representation of an individual registry for each data type to consider;
- **Request Overhead:** The model's calculated request overhead is measured as the time, in milliseconds, between the sending of the API request and the receiving of the system's answer.

## 5.2 Quantification of Evaluation Criteria

In order to quantify the metrics defined in the previous section, a formula was devised in order to translate the performed evaluation into single value, representing the model's suitability for Citibrain's IoT platform. The general formula of the data model's quality (MQ) is represented by:

$$MQ = W_{Qt} \cdot QtEv + W_{Ql} \cdot QlEv$$

Where $W_{Qt}$ and $W_{Ql}$ represent the weights, and QtEv and QlEv represent the measures of the quantitative and qualitative evaluations. In this thesis, and without loss of generality, it was chosen to give the quantitative evaluation a higher weight, because it is more objectively measurable. Thus, throughout the rest of this work $W_{Qt}$ is given a value of 0.7 and $W_{Ql}$ a value of 0.3.

### 5.2.1 Quantitative Metrics

The Quantitative Evaluation is calculated with the values of the memory efficiency (M.E.), temporal efficiency (T.E.), representational efficiency (R.E.) and request efficiency (Rq.E.), introduced in the conversion of the data from the base model to the new data model.

$$QtEv = W_{ME} \cdot M.E. + W_{TE} \cdot T.E. + W_{RE} \cdot R.E. + W_{RqE} \cdot Rq.E.$$

As all the efficiency values can be measured with the same precision, and all affect the total overhead in the same manner, the same weight should be attributed to all of them - 0.25.

Since the final values of the formula vary between 0 and 1, with 1 representing the most suitable model possible, the normalized values of the overheads are subtracted from 1, thus representing the data models' efficiency.

For the measurement of the memory, temporal and request overheads, ten measurements are made for the conversion of the entirety of the database registries of each vertical solution. An average of the ten values is then calculated, and the results of all vertical solutions' mean times are added. The representational overhead is calculated by summing the sizes of all JSON objects provided by the API for a request of the bulk of the data contained in the database. In order to normalize the values, the total overhead of each metric for each model is divided by the sum of all model's corresponding overheads and then subtracted from 1. The calculation of the representational overhead can be consulted in the following formula:

$$R.O.^{Model\ X} = 1 - \Sigma\ (Size\ (JSONObjects^{ModelX})) / \Sigma\ (JSONObjects)$$

### 5.2.2 Qualitative Metrics

The qualitative evaluation, represents the quantification of the more difficultly measured characteristics that define the quality of a data model. Within this group of criteria, two sub-groups can be identified: one containing the more easily measurable metrics, designated as Objective Qualitative Criteria (Obj), and one containing the more abstract metrics, designated as the Subjective Qualitative Criteria (Subj):

$$QlEv = W_{Obj} \cdot Obj + W_{Subj} \cdot Subj$$

Where $W_{Obj}$ and $W_{Subj}$ represent the weights attributed to each set of metrics. For this thesis, following the same principle and the weight distribution between the qualitative and the quantitative analysis, the objective analysis was attributed a bigger weight, of 0.75, and the subjective analysis the smaller (0.25).

The objective qualitative criteria is defined by six metrics: (i) the amount of available documentation and support (Doc), (ii) the model's overlap with Citibrain's current information structure (Ovrlp), (iii) the simplicity of the model's implementation (Simp), (iv) the model's completeness (Comp), (v) the model's implementability (Imp), and (vi) the existence of implementation tools (ITools):

$$Obj = W_{Doc} \cdot Doc + W_{Ovrlp} \cdot Ovrlp + W_{Simp} \cdot Simp + W_{Comp} \cdot Comp + W_{Impl} \cdot Impl + W_{IT} \cdot ITools$$

Where $W_{metric}$ represents the weight of each metric. For the evaluation present in this thesis, all the metrics were attributed the same weight, of 1/6.

The documentation value is measured through a point system, where a point is attributed to the model if it satisfies each of the questions asked in the previous section regarding documentation and available support:

1. Does the platform provide a specification for its data model?
2. Do implementation guidelines exist?
3. Are there any tutorials provided?
4. Does a repository exist?
5. Are examples given of the models' implementation?
6. Does a community exist? If so, is it responsive?
7. Are there any questions related to the models on *Stack Overflow?* Do they have accepted answers?

After calculating the value for all the studied models, the result is normalized, by diving it by the maximum possible value - 7.

The simplicity value is calculated by adding the number of entities in the models and the relations, and dividing the result by the maximum value observed. The value is then subtracted from 1.

The completeness value is calculated by dividing the sum of entities and relations of the model by the sum of entities and relations of the original base model, and dividing the result by the maximum value of all the models. The outcome is then subtracted from 1.

The implementability value is calculated by dividing the number of hours spent implementing each of the models, by the total number of hours spent in the implementation of all the models. Since the model with the least implementation hours is the most desirable choice, the result is also subtracted from 1.

The implementation tools value is calculated in similar fashion to the documentation metric. The model gets one point for each question asked in the previous section it can satisfy, and is then normalized.

The overlap with current resident models is calculated by mapping the base model's entities to the data models'. The result is then normalized.

Regarding the subjective criteria, three metrics were chosen: the easiness of implementation (EoI), the understandability (Undst), and the flexibility (Flex) of the model, with the final value calculated as is shown in the following formula:

$$Subj = W_{EoI} \cdot EoI + W_{Undst} \cdot Undst + W_{Flex} \cdot Flex$$

For the developed analysis in this thesis, all the subjective criteria were given the same weight of 1/3.

Due to the highly subjective nature of these three metrics, the values were obtained through the elaboration of a Likert type survey, consisting of three questions, with five possible answers, ranging from Strongly Disagree to Strongly Agree:

1. The Data Model is easy to implement
2. The Data Model's concepts and entities are easy to understand
3. The Data Model can be considered flexible to changes in the database

The possible answers for these statements were (i) Strongly Agree (with a value of five points), (ii) Agree (with a value of four points), (iii) Neither Agree or Disagree (with a value of three points), (iv) Disagree (with a value of two points), and (v) Strongly Disagree (with a value of one point). The answer's values are then added and divided by the product of the number of questions and the maximum possible value of the answer (15, in this case) [24].

### 5.2.3 Data Model Overall Quality

By combining the entirety of the criteria described in the previous sections, the full extent of the formula can then be calculated as follows:

$$M.Q = W_{Qt} \cdot (W_{ME} \cdot M.E + W_{TE} \cdot T.E. + W_{RE} \cdot R.E. + W_{Rq.E.} \cdot Rq.E.) + W_{Ql} \cdot (W_{Obj} \cdot (W_{Ovrlp} \cdot Ovrlp + W_{Doc} \cdot Doc + W_{Simp} \cdot Simp + W_{Comp} \cdot Comp + W_{Imp} \cdot Imp + W_{ITools} \cdot ITools) + W_{Subj} \cdot (W_{EoI} \cdot EoI + W_{Undst} \cdot Undst + W_{Flex} \cdot Flex))$$

Applying the chosen weights for the performed evaluation in this thesis, the final value of the model's quality can be calculated as follows:

$$M.Q = 0.7\ (1/4 \cdot M.E + 1/4 \cdot T.E. + 1/4 \cdot R.E. + 1/4 \cdot Rq.E.) + 0.3 \cdot (0.75 \cdot (1/6 \cdot Ovrlp + 1/6 \cdot Doc + 1/6 \cdot Simp + 1/6 \cdot Comp + 1/6 \cdot Imp + 1/6 \cdot ITools) + 0.25 \cdot (1/3 \cdot EoI + 1/3 \cdot Undst + 1/3 \cdot Flex))$$

For a better understanding and easier analysis of the chosen criteria, Table 4 summarizes the defined metrics and respective quantification methodology.

**Table 4: Data Model Evaluation Framework Metrics and Quantification**

| Analysis | Criteria | Description | |
|---|---|---|---|
| **Quantitative** | Temporal Overhead | Time spent in conversion / Total | |
| | Memory Overhead | Computer resources used in conversion / Total | |
| | Representational Overhead | Total size of information sample / Total | |
| | Request Overhead | Time elapsed between the dispatch of the HTTP request and the reception of the response / Total | |
| **Qualitative** Objective | Documentation And Support | Specification | 1 point per metric / 7 |
| | | Guidelines | |
| | | Tutorial | |
| | | Repository | |
| | | StackOverflow | |
| | | Community | |
| | | Examples | |
| | Simplicity | $1 - ER_{model} / Max(ER)$[31] | |
| | Completeness | $1 - (ER_{model} / ER_{base\ model})/Max_{completeness}$ | |
| | Implementability | 1 – (Number of implementation hours) / Total | |
| | Overlap | 1 point per entity / base model entities | |
| | Implementation Tools | SDK | 1 point per metric / 3 |
| | | Libraries | |
| | | Validation Tools | |
| **Qualitative** Subjective | Flexibility | 5 Step Likert-style scale item Ranging from “Strongly Agree” to “Strongly Disagree” | |
| | Easiness of Implementation | | |
| | Understandability | | |

[31] E.R.: The sum of number of entities and relations between them, as defined by the generalized data model.

## 5.3 Final Considerations on the Evaluation

Regarding the quantitative analysis, all the measurements were made in a dedicated DS2 Microsoft Azure Ubuntu virtual machine[32], with the following specifications:

- 2 CPU cores
- 7 GiBs of RAM;
- 100 GBs of disk space;
- local SSD drive with 14 GiB;

The values of the temporal overheads, and subsequently efficiency, were measured by the Python function *TimeIt*[33], responsible for measuring the execution time of code snippets. The values of the memory overheads (and subsequently efficiency) were measured through the use of Python *resource.usage(ru_maxrss)*[34] function, responsible for measuring the resource usage information, in this case, the maximum resident set size.

In spite of the devised framework being designed to be a generic approach to the ranking and evaluation of data models for the IoT, it was designed with the notion of the limitations provided by the size of the test group, which consisted of only one subject. With these notions in mind, some considerations can be made regarding the attributed weights and the choice of the metrics:

- The metrics involving the number of hours of implementation, and the subjective measurements should be evaluated by more than one developer, in order to make the results more general. However, there was a lack of means to set up a study in such conditions, in this work.
- If the test group is comprised of several subjects with experience in data modelling and data model evaluation, different weights might be attributed to both the quantitative and qualitative analysis, reaching up to the same weight.
- Some other Likert Items can be included in the subjective portion of the formula, in order to attain a more complete idea of the developers' experience with the data model. If this is the case, in order to avoid *respondent fatigue*, an even scale should be used. This was not the case with the conducted study, due to the small amount of items present in the developed survey.

[32] docs.microsoft.com/pt-pt/azure/virtual-machines/windows/sizes-general
[33] docs.python.org/2/library/timeit.html
[34] docs.python.org/2/library/resource.html

Although the objective metrics of the formula should always have a bigger impact in the final value, given the easily measurable nature, it is always important to have in mind that, as stated by Maier [21], the quality of the data modelling experience provides an equally important notion as quality of the data model itself. This last notion is described by the author as a "method, which supports the process of adaptation, standardization and integration in the development of application systems".

# Chapter 6

# Data Models: A Deeper Analysis

This chapter contains a more detailed evaluation of the five data models selected in Chapter 4 - FIWARE, OGC SensorThings, W3C Generic Sensor, IPSO Smart Objects, and oneIoTa - regarding their suitability for Citibrain's smart city platform. The evaluation is made using the criteria selected in the previous chapter, making use of the devised weighted formula. For easier readability and standalone analysis of each models' performance, the evaluation presented in this chapter is divided by data model, with each section following an identical text structure. Each evaluation proceeds as follows:

1. A qualitative analysis is performed, evaluating both objective and subjective aspects of the model. In a first step, the former aspects of the model are assessed, comprised of entity similarity, amount of documentation and support, simplicity, completeness, implementability, and the existence of implementation tools. In a second step, more abstract metrics are analyzed - understandability, easiness of development and flexibility.

2. A quantitative analysis is carried out, assessing metrics of memory, temporal, representational and request overheads.

Regarding the quantitative analysis, the following sample was used for each vertical solution, following the entities described in chapter 3:

- **Environment Management**: 19 instances of assets, and 9 events;

- **Parking Management**: 1 Parking Area, 8 parking spots, 22 Assets and a maximum of 500 events per Asset;
- **Traffic Management**: 11 Assets, and a maximum of 500 measurements per Asset;
- **Waste Management**: 4 Waste Areas, 14 Waste Containers, 23 Assets and 11 Waste Events.

Regarding the qualitative analysis, for the metrics of simplicity and completeness for the instantiated models, the worst case scenario was used, i.e., the one presenting more entities and relations.

Upon performing the evaluation of the different models which takes place in the next sections, the present chapter concludes with a comparison of the results, and with the selection of the model representing the best fit for Citibrain. The entirety of the performed measures can be consulted in annex B.

## 6.1 FIWARE Data Model

For the tests presented in the next subsections, the implemented core entities of the FIWARE data model are the ones presented in Figure 22.

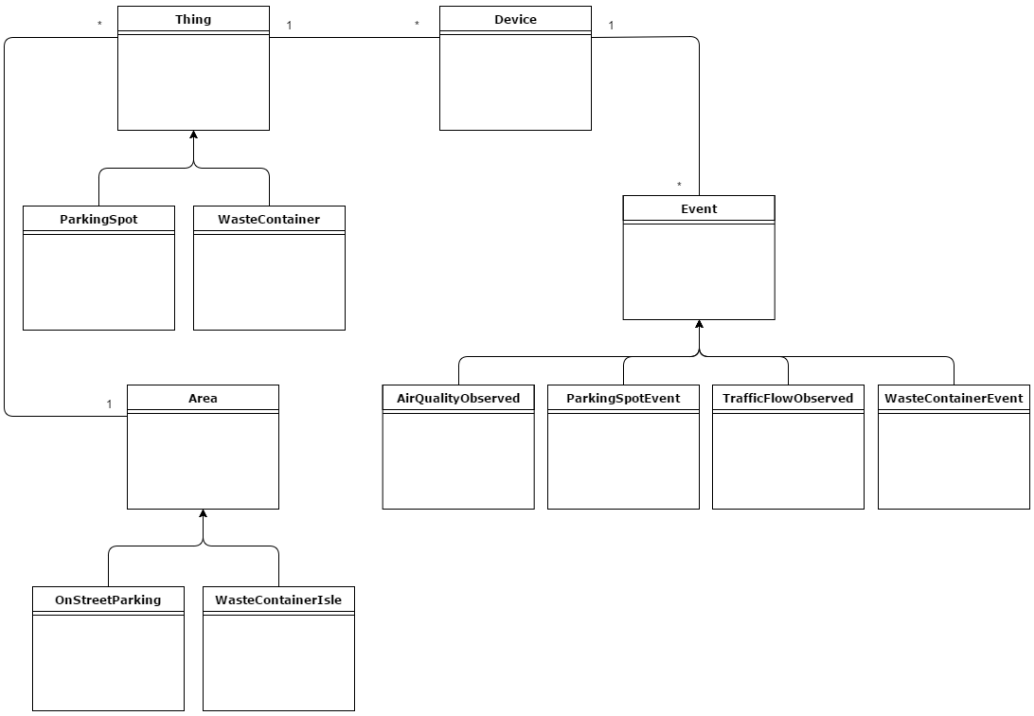



**Figure 22: Fiware's implemented core entities**

### 6.1.1 Qualitative Analysis

Regarding the amount of available documentation and support, FIWARE was able to satisfy all defined criteria, presenting the following results:

- **Specification**: FIWARE disposes of a specification for each data model, contained in the implementation guidelines;
- **Guidelines**: Implementation guidelines for each data model were found in the official website[35], composed of a brief introduction and a detailed description of each model and its fields;
- **Tutorials:** FIWARE disposes of a YouTube channel, containing a playlist of 26 video tutorials [36];
- **Examples:** Data model implementation examples are provided in the guidelines for each data model[37];
- **Repository:** FIWARE has a Github repository[38] containing documentation and data model implementations in Python language, and in CSV format;
- **Online Community:** FIWARE disposes of its own online community, FIWARE Q&A,[39] containing 172 questions, with only 4 unanswered, giving the platform a total of 97.67% of responsiveness;
- **StackOverflow presence:** There are 727 questions on StackOverflow[40], tagged with 'FIWARE', with 218 marked as unanswered, giving the community a responsiveness of 70.01% responsiveness.

Concerning entity similarity, as classified in section 8 of Chapter 4, FIWARE provides instantiated models for all of Citibrain's vertical solutions entities and relations.

Defining a total of 5 entities and 4 relations between them, FIWARE scored a total of 9 in the simplicity metric, and, when compared with Citibrain's base model, which defines 4 entities and 4 relations between them, FIWARE obtained a final value of 1.125.

Respecting implementability, for the development of the FIWARE data structure and data conversion software, a total of 16.67 hours were spent, distributed as follows: (i) 0.85 hours for

[35] www.fiware-datamodels.readthedocs.io/en/latest/index.html (Accessed May 29th, 2017)
[36] www.youtube.com/playlist?list=PLR9elAI9JscSOuSnwIkGzSVW1QKgfDk6d (Accessed May 18th, 2017)
[37] www.fiware-datamodels.readthedocs.io/en/latest/Device/Device/doc/spec/index.html (Accessed May 29, 2017)
[38] www.fiware-datamodels .readthedocs.io/en/latest/index.html (Accessed May 29, 2017)
[39] ask.fiware.org/questions/ (Accessed May 29, 2017)
[40] As of May 18, 2017

the implementation of the environment management data models, (ii) 2.18 hours for the implementation of the smart parking management data models, (iii) 1.85 hours for the implementation of the traffic management data models, and (iv) 11.5 hours for the implementation of the air quality management data models.

Regarding implementation tools, FIWARE was able to satisfy all of the defined criteria, offering an SDK, composed of several libraries and tools for the implementation of the platform, providing a Node.js client library for the Orion Context Broker, and a FIWARE Validator[41].

In the subjective qualitative analysis, FIWARE scored a total of 14 points, distributed as follows:

- **Easiness of Implementation**: 5 points, having a choice of 'Strongly Agree', in the 'The data model is easy to implement' Likert Item
- **Understandability**: 5 points, having a choice of 'Strongly Agree' in the 'The data model's concepts are easy to understand' Likert Item.
- **Flexibility**: 4 points, having a choice of 'Agree' in the 'The data model is flexible for changes in the database structure' Likert Item.

### 6.1.2 Quantitative Analysis

After the conversion of Citibrain base model's data into FIWARE's structure, a total of 11 575 model instances were created: 9 *AirQualityObserved*, 1 *OnStreetParking*, 8 *ParkingSpot*, 33 *Device*, 6 003 *ParkingSpotEvent*, 5 489 *TrafficFlowObserved*, 5 *WasteContainerModels*, 14 *WasteContainer*¸ 2 *WasteContainerIsle,* and 11 *WasteContainerEvent.*

For the temporal overhead measurements, Fiware scored a total average of 69.303 seconds, distributed in the following way:

- **Smart Environment Management**: 0.28 seconds, with a standard deviation of 0.0059 seconds
- **Smart Parking Management:** 30.69 seconds, with a standard deviation of 0.79 seconds
- **Smart Traffic Management:** 35.63 seconds, with a standard deviation of 0.52
- **Smart Waste Management:** 0.54 seconds with a standard deviation of 0.046 seconds.

Regarding the memory overhead measurements, Fiware scored a total value of 2 910.8 kilobytes of memory used, distributed among the four vertical solutions in the following way:

[41] www.github.com/ging/fiware-validator (Accessed May 18, 2017)

- **Smart Environment Management:** 281.6 KBs, with a standard deviation of 4.30 KBs;
- **Smart Parking Management:** 1 601.6 KBs, with a standard deviation of 52.40 KBs;
- **Smart Traffic Management:** 903.2 KBs, with a standard deviation of 53.89 KBs;
- **Smart Waste Management:** 124.4 KBs, with a standard deviation of 1.27 KBs.

Concerning the representational overheads, FIWARE had the following values for its seven entities:

- ***Device***: 4.51 KBs;
- ***AirQualityObserved***: 4.61 KBs;
- ***OnStreetParking***: 1.48 KBs;
- ***ParkingSpot***: 2.96 KBs;
- ***TrafficFlowObserved***: 2 665.62 KBs;
- ***WasteContainer***: 8.12 KBs;
- ***WasteContainerIsle***: 8.121 KBs;
- ***WasteContainerModel***: 645 KBs.

For the Request Overhead measurements, FIWARE scored a total of 9.45 seconds, distributed among the implemented data models in the following way: (i) 1 873 ms for the *Device* models, (ii) 842 ms for the *AirQualityObserved* models, (iii) 777 ms for the *OnStreetParking* models, (iv) 628 ms for the *ParkingSpot* models, (v) 3 314 ms for the *TrafficFlowObserved* models, (vi) 757 ms for the *WasteContainer* models, (vii) 627 ms for the *WasteContainerIsle* models, and (viii) 628 ms for the *WasteContainerModel* models.

After applying FIWARE's measurement values to the weighted formula, the data models scored a final value of 0.86.

### 6.1.3 Other Considerations

Fiware presents one big limitation, regarding the data representation of the Parking and Waste Management vertical solutions. Since the data model incorporates the last measurements in the parking spots and waste containers models respectively, it is only capable of representing said measurements, not providing a sensor reading data model. In order to provide several

measurements within the same entity, one would have to provide several identic data models of the same entities (Parking Spot and Waste Container), overloading the database with duplicated information (only the measurement values would vary between the files). The solution for the aforementioned problem was the creation of two new data models, *ParkingSpotEvent* and *WasteContainerEvent*, capable of representing the measurements captured by the devices over times.

Besides the mentioned problem, Fiware achieved good results in the framework's evaluation regarding the overheads, being the most well documented of the models. With Citibrain's current data model being based in Fiware's architecture, the model can be considered a good candidate.

## 6.2 Generic Sensor Data Model

For the implementation of W3C's data model, some architectural modifications were made, in order to approach its structure to Citibrain's. In the figure bellow, the implementation architecture can be observed:

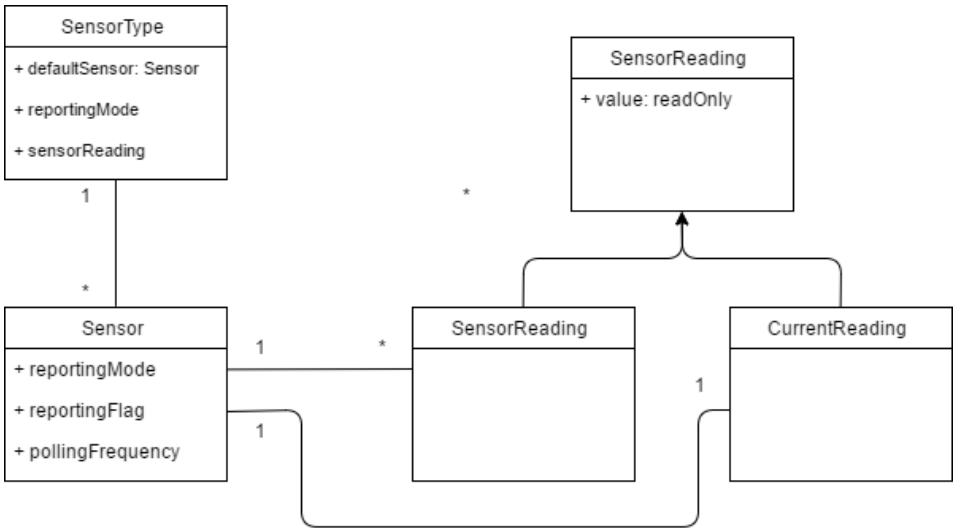



**Figure 23: Generic Sensor Implemented Core Entities**

Differing from the original architecture, two main aspects can be identified:

- The creation of an abstract class, *Reading*, which is instantiated by both *SensorReading* and *CurrentReading;*
- The exclusion of the class *DefaultSensor*.

### 6.2.1 Qualitative Analysis

Regarding the qualitative evaluation, Generic Sensor was able to satisfy five of the seven metrics defined for the documentation and support analysis, distributed in the following way:

- **Specification**: Generic Sensor disposes of a specification in the W3C's official Github repository[42];
- **Guidelines**: Generic Sensor provides no guidelines document, having however one use case document to be available soon [43];
- **Tutorials**: Generic Sensor provides no tutorials either for the implementation of the API nor the implementation of the data models [22];
- **Examples**: Two web-app examples are provided in W3C's official Github repository[21];
- **Repository**: W3C has an official Github repository [21];
- **Online Community**: W3C has an official Community, Sensor Web, with the last post being form 2013, and offering no Question and Answers section [22], thus scoring a value of 0 in the criterion;
- **StackOverflow presence**: Generic Sensor API has a total of 4 questions tagged in StackOverflow, with two unanswered, giving the platform's StackOverflow presence a value of 50%;

Regarding entity similarity, and according to section 4.8, Generic Sensor is able to represent five of Citibrain's nine total entities, not being able to represent the ones referring to Things (*ParkingSpot* and *WasteContainer)*, and the ones referring to Areas (*ParkingArea* and *WasteArea*).

Concerning simplicity, and using the original model's architecture, Generic Sensor defines a total of four entities and three relations, scoring a total of 7 in the simplicity metric. Using the same value, and comparing it to Citibrain's original base model's value of 8, Generic Sensor scores a total of 0.875 in the Completeness metric.

Regarding implementability, a total of 6.93 hours was spent in the development of the conversion software, distributed as follows: (i) 1.5 hours for the implementation of the model's general architecture, (ii) 2.4 hours for the development of the environment management solution, (ii) 1.47 hours for the development of the smart parking solution, (iii) 0.57 hours for the

[42] w3c.github.io/sensors/ (accessed May 19, 2017)
[43] As of May 19, 2017

development of the traffic management solution, and (iv) 1 hour the development of the smart waste management solution.

Finally, referring to the existence of implementation tools, SensorThings was able to satisfy one of the three imposed metrics:

- **Existence of SDKs**: W3C offers no SDK for the implementation of the platform [22];
- **Libraries**: There are no available libraries for the platform's implementation [22];
- **Validation Tools**: W3C offers one testing tool for the platform, available in the official Github repository [23].

In respect to the subjective portion of the qualitative analysis, Generic Sensor's results were the following:

- **Easiness of Implementation**: 5 points, having a choice of 'Strongly Agree', in the 'The data model is easy to implement' Likert Item. Due to the data model's simplicity and low number of entities the implementation process was one of the shortest ones;
- **Understandability**: 5 points, having a choice of 'Strongly agree' in the 'The Data Model's concepts and entities are easy to understand' Likert Item. Due, once again, to the simplistic and minimalist nature of the data model, all the defined concepts were simple to understand;
- **Flexibility**: 5 points, having a choice of 'Strongly Agree' in the 'The Data Model can be considered flexible to changes in the original information structure'. Due to the small number of entities the model identifies, changes in the original base model are almost meaningless.

### 6.2.2 Quantitative Analysis

After the conversion of Citibrain base model's data into Generic Sensor's structure, an additional 11 889 database registries were created: 18 *SensorType* instances, 280 *Sensor* instances, and 11 591 *Reading* instances.

For the temporal overhead measurements, Generic Sensor scored a total of 82.02 seconds, distributed in the following way:

- **Smart Environment Management:** 2.30 seconds with a standard deviation of 0.041 seconds;
- **Smart Parking Management:** 36.84 seconds with a standard deviation of 1.50 seconds;

- **Smart Traffic Management:** 38.14 seconds with a standard deviation of 0.89 seconds;
- **Smart Waste Management:** 0.62 seconds with a standard deviation of 0.030 seconds;

Regarding the memory overhead measurements, Fiware scored and average of 3 891.2 KBs of memory used, distributed among Citibrain's four vertical solutions in the following way:

- **Smart Environment Management:** 939.2 KBs, with a standard deviation of 75.56 KBs;
- **Smart Parking Management:** 1 417.2 KBs, with a standard deviation of 99.97 KBs;
- **Smart Traffic Management:** 1 158.8 KBs, with a standard deviation of 53.60 KBs;
- **Smart Waste Management:** 376 KBs, with a standard deviation of 25.30 KBs.

Concerning the representational overhead, Generic Sensor, presented the following file sizes, totaling an overall size of 2 670.34 KBs:

- **SensorType**: 11.93 KBs;
- **Sensor**: 405.59 KBs;
- **Reading**: 2 252.82 KBs;

Finally, in respect to the request overhead measurements, Generic Sensor scored a total of 13363 seconds, distributed in the following way: (i) 1 263 ms for the query of the entirety of the *SensorType* models, (ii) 1 735 ms for the entirety of the *Sensor* models, and (iii) 10 365 for the entirety of the *Reading* models. Upon introducing all the values into the weighted formula, Generic Sensor scored a final model quality value of 0.78.

## 6.3 IPSO Smart Objects Data Model

For the implementation of IPSO Smart Object's data model, and based upon the Object Resource Model described in section 4 of chapter 4, the following architecture was used:

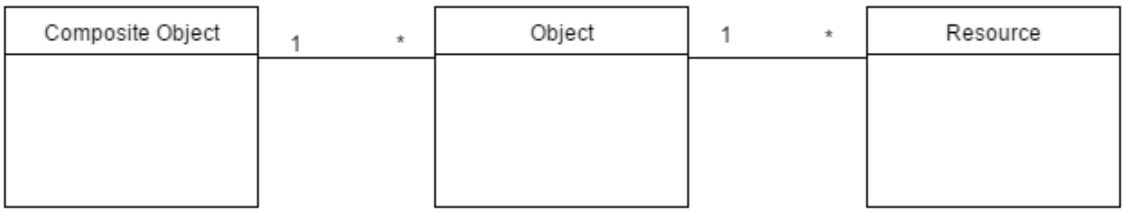



**Figure 24: IPSO Smart Objects' Implemented Core Entities**

### 6.3.1 Qualitative Analysis

Regarding the amount of available documentation and support, IPSO Smart Objects was able to satisfy all defined criteria, presenting the following results:

- **Specification**: Smart Objects disposes of a specification in the form of a scientific paper [35];
- **Guidelines**: The specification of the data model contains guidelines for its implementation [37];
- **Tutorials:** In the research activities, there were no tutorials found for the implementation of IPSO Smart Objects[44];
- **Examples:** The data model specification contains examples of object definitions;
- **Repository:** IPSO Smart Objects possesses an official Github repository, however containing only the aforementioned Specification and Guideline papers[23];
- **Online Community:** A community for IPSO Smart Object developers exists, but its access is limited to IPSO Alliance members, making the data model score a value of 0 in this criteria [23];
- **StackOverflow presence:** There are a total of 33 questions on StackOverflow tagged with 'IPSO Smart Objects', with only 6 unanswered, giving the StackOverflow community a score of 81.82% responsiveness.

Concerning entity similarity, as classified in section 8 of Chapter 4, IPSO Smart Objects is able to represent all Citibrain's 9 entities.

Defining a total of 3 entities and 2 relations between them, Smart Objects scored a total of 5 in the simplicity metric, and, when compared with Citibrain's base model, which defines 4 entities and 4 relations between them, the data model obtained a final value of 0.625.

[44] As of May 29, 2017

Respecting implementability, for the development of the data structure and data conversion software, a total of 6.67 hours were spent, distributed as follows: (i) 1.18 hours for the implementation of the model's general architecture, (ii) 1.73 hours for the implementation of the environment management data models, (iii) 1.17 hours for the implementation of the smart parking management data models, (iv) 0.55 hours for the implementation of the traffic management data models, and (iv) 2.04 hours for the implementation of the air quality management data models.

Regarding implementation tools, IPSO was able to satisfy all of the defined criteria, offering an SDK, Wakaama[45], at least two libraries – leshan[46] and lwm2m-node-lib[47]-, and a Firefox add-on for data validation, OMA LWM2M DevKit[48].

In the subjective qualitative analysis, IPSO Smart Objects scored a total of 12 points, distributed as follows:

- **Easiness of Implementation**: 3 points, having a choice of 'Neither Agree or Disagree', in the 'The data model is easy to implement' Likert Item;
- **Understandability**: 4 points, having a choice of 'Agree' in the 'The data model's concepts are easy to understand' Likert Item;
- **Flexibility**: 5 points, having a choice of 'Strongly Agree' in the 'The data model is flexible for changes in the database structure' Likert Item.

### 6.3.2 Quantitative Analysis

After the conversion of Citibrain's base model's data into IPSO Smart Object's structure, an additional 11 659 database registries were created: 52 *CompositeObject* instances, 198 *Object* instances, and 11 409 *Resource* models.

For the temporal overhead measurements, Smart Objects scored a total average of 150.0264884 seconds, distributed in the following way:

- **Smart Environment Management:** 2.46 seconds with a standard deviation of 0.080 seconds;
- **Smart Parking Management:** 75.11 seconds with a standard deviation of 1.13 seconds;
- **Smart Traffic Management:** 68.43 seconds with a standard deviation of 0.834 seconds;

[45] www.github.com/eclipse/wakaama (Accessed May 29, 2017)
[46] www.github.com/eclipse/leshan (Accessed May 29, 2017)
[47] www.github.com/telefonicaid/lwm2m-node-lib (Accessed May 29, 2017)
[48] www.addons.mozilla.org/en-US/firefox/addon/oma-lwm2m-devkit/ (Accessed May 29, 2017)

- **Smart Waste Management:** 1.23 seconds with a standard deviation of 0.030 seconds.

Regarding the memory overhead measurements, IPSO Smart Objects scored a total value of 4335.2 kilobytes of maximum memory resources used, distributed among the four vertical solutions in the following way:

- **Smart Environment Management:** 603.6 KBs, with a standard deviation of 63.73 KBs;
- **Smart Parking Management:** 1 481.2 KBs, with a standard deviation of 70.53 KBs;
- **Smart Traffic Management:** 1 866.4 KBs, with a standard deviation of 72.54 KBs;
- **Smart Waste Management:** 384 KBs, with a standard deviation of 0 KBs.

Concerning the representational overheads, the data model achieved the following values for its three defined entities, totaling a value of 1 988.83 KBs.

- **Composite Object**: 21.52 KBs;
- **Object**: 507.81 KBs;
- **Resource**: 1 459.50 KBs

For the Request Overhead measurements, Smart Objects scored a total of 4.797 seconds, the lowest value of all the models, distributed among the implemented data models in the following way: (i) 1 813 ms for the *Composite Object* instances, (ii) 1 469 ms for the *Object* instances, (iii) 1 515 ms for the *Resource* models.

Upon the application of the weighted formula on the measured and calculated values, IPSO Smart Objects got a total model quality value of 0.83.

## 6.4 oneIoTa Data Model

For the tests presented in the next subsections, the implemented core entities of the oneIoTa data model are the ones presented in the figure below.

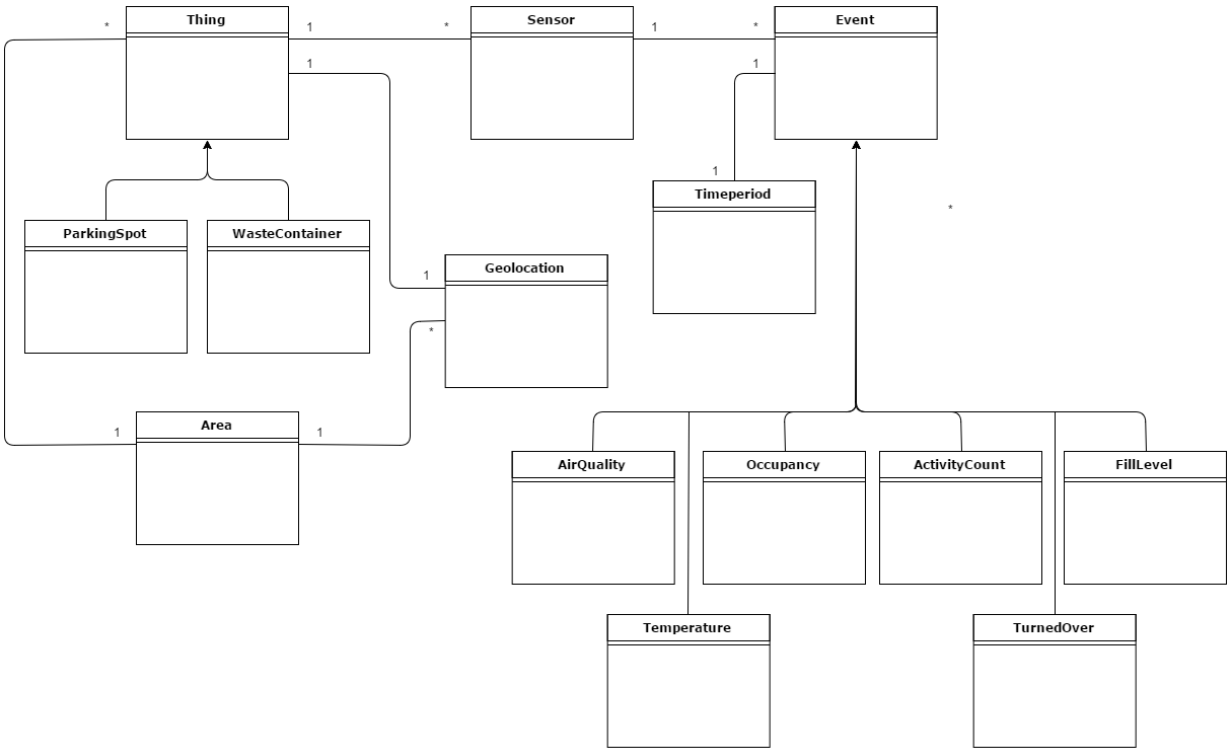



**Figure 25: oneIoTa Implemented Core Entities**

The entities *Thing* and *Event* are abstract, and are only represent to ease the readability of the chart and thus were not implemented.

### 6.4.1 Qualitative Analysis

oneIoTa was the least documented of the studied models, having been able to satisfy only two of the seven proposed metrics[49]:

- **Specification**: oneIoTa provides no specification for its data models;
- **Guidelines**: No guidelines were found for oneIoTa;
- **Tutorials:** No tutorials are provided by the data model tool;
- **Examples:** oneIoTa offers implementation examples of all its 232 models, both in JSON and RAML formats;

[49] As of April 4th, 2017

- **Repository:** The data model tool possesses one official Github repository, containing the implementation examples mentioned in the previous metric[50];

- **Online Community:** An online community for the data model tools exists, being responsible for the approval of new data model submissions. However, no questions and answers forum was found;

- **StackOverflow presence:** No questions tagged with 'oneIoTa' were found on StackOverflow.

Concerning entity similarity, as classified in section 8 of Chapter 4, oneIoTa is capable of representing all Citibrain's vertical solutions entities and relations. Defining a total of 6 entities and 7 relations between them, oneIoTa scored a total of 13 in the simplicity metric, and, when compared with Citibrain's base model, which defines 4 entities and 4 relations between them, the data model obtained a final value of 1.625.

Respecting implementability, for the development of oneIoTa's data structure and data conversion software, a total of 21.58 hours were spent, distributed as follows: (i) 3.5 hours for the implementation of the model's general architecture, (ii) 3 hours for the implementation of the environment management data models, (iii) 4.58 hours for the implementation of the smart parking management data models, (iv) 4.03 hours for the implementation of the traffic management data models, and (iv) 6.47 hours for the implementation of the air quality management data models.

Regarding implementation tools, oneIota was able to satisfy only one of the defined criteria, offering an SDK, an API designer console[51].

In the subjective qualitative analysis, oneIoTa scored a total of 13 points, distributed as follows:

- **Easiness of Implementation**: 4 points, having a choice of 'Agree', in the 'The data model is easy to implement' Likert Item;

- **Understandability**: 5 points, having a choice of 'Strongly Agree' in the 'The data model's concepts are easy to understand' Likert Item;

- **Flexibility**: 4 points, having a choice of 'Agree' in the 'The data model is flexible for changes in the database structure' Likert Item.

[50] www.oneiota.org (Accessed ay 30th, 2017)
[51] www.github.com/OpenInterConnect/api-designer (Accessed May 30th, 2017)

### 6.4.2 Quantitative Analysis

After the conversion of Citibrain base model's data into oneIoTa's structure, a total of 13 741 model instances were created: 8 080 *TimePeriod*, 25 *Geolocation*, 27 *Sensor*, 51 *AirQuality*, 3 *Area*, 8 *ParkingSpot*, 5 500 *ActivityCount*, 14 *WasteContainer*¸ 11 *Temperature*, 11 *FillLevel* and 11 *TurnedOver*.

For the temporal overhead measurements, the data model scored a total average of 117.3461461 seconds, distributed in the following way:

- **Smart Environment Management:** 0.58 seconds, with a standard deviation of 0.020 seconds;
- **Smart Parking Management:** 50.50 seconds, with a standard deviation of 0.42 seconds;
- **Smart Traffic Management:** 65.24 seconds, with a standard deviation of 0.57 seconds;
- **Smart Environment Management:** 0.92 seconds, with a standard deviation of 0.021 seconds.

Regarding the memory overhead measurements, oneIoTa scored a total value of 5894.4 kilobytes of memory used, distributed among the four vertical solutions in the following way:

- **Smart Environment Management:** 643.2 KBs, with a standard deviation of 46.91 KBs;
- **Smart Parking Management:** 3 126 KBs, with a standard deviation of 66.67 KBs;
- **Smart Traffic Management:** 1 209.2 KBs, with a standard deviation of 94.83 KBs;
- **Smart Waste Management:** 916 KBs, with a standard deviation of 64.36 KBs.

Concerning the representational overheads, oneIoTa had the following values for its twelve entities:

- **TimePeriod**: 12 368.19 KBs;
- **TurnedOver**: 16.07 KBs;
- **WasteContainer**: 22 899 KBs;
- **Temperature**: 16.04 KBs;

- **ActivityCount**: 7 364.83 KBs;
- **AirQuality**: 172.06 KBs;
- **FillLevel**: 16.17 KBs;
- **Occupancy**: 4 389.15 KBs;
- **ParkingSpot**: 140.79 KBs;
- **Sensor**: 69.91 KBs;
- **Geolocation**: 45.97 KBs;
- **Area**: 4.37 KBs;
- **AreaMultipolygon**: 4.73 KBs.

For the Request Overhead measurements, oneIoTa scored a total of 27.22 seconds, distributed among the implemented data models in the following way: (i) 10 114ms for the *TimePeriod* models, (ii) 654 ms for the *TurnedOver* models, (iii) 693 ms for the *WasteContainer* models, (iv) 633 ms for the *Temperature* models, (v) 4 611 ms for the *ActivityCount* models, (vi) 3 947 ms for the *AirQuality* models, (vii) 695 ms for the *FillLevel* models, (viii) 1 631 ms for the *Occupancy* models, (ix) 859 ms for the *ParkingSpot* models, (x) 924 ms for the *Sensor* models, (xi) 948 ms for the *Geolocation* models, (xii) 728 ms for the *Area* models, and (xiii) 787 for the *AreaMultipolygon* models.

After all the values were put through the weighted formula, oneIoTa scored the lowest total value of all the analyzed data models: 0.58.

## 6.5 SensorThings Data Model

For the implementation of SensorThings data model, no changes were made to the original architecture, having the developed solution the same entities and relations presented in Figure 21, in Chapter 4.

### 6.5.1 Qualitative Analysis

Regarding the amount of available documentation and support, SensorThings was able to satisfy five of the seven defined criteria, presenting the following results:

- **Specification**: SensorThings disposes of a comprehensive specification for both the API itself and its data model[52];
- **Guidelines**: Guidelines for the implementation of the API are available in the specification[31];
- **Tutorials:** SensorThings provides access to tutorials to its users[53];
- **Examples:** There were no examples found for the implementation of SensorThings[54];
- **Repository:** SensorThings has an official Github repository [55];
- **Online Community:** During the research activities, there was no online questions and answers forum found for the standard[33];
- **StackOverflow presence:** There were 9 results tagged with 'SensorThings' found in StackOverflow, with 3 unanswered questions, giving the community a responsiveness value of 66.67%.

Concerning entity similarity, as classified in section 8 of Chapter 4, SensorThings data model is capable of representing 7 of the 9 Citibrain's vertical solutions entities.

Defining a total of 8 entities and 8 relations between them, SensorThings scored the highest value of all data models in the simplicity metric, with a total of 16. When compared with Citibrain's base model, which defines 4 entities and 4 relations between them, the data model obtained a score of 2, also the highest value of all data models.

Respecting implementability, for the development of SensorThings' data structure and data conversion software, a total of 32.2 hours were spent, distributed in the following fashion: (i) 4.3 hours for the implementation of the model's general architecture, (ii) 4.72 hours for the implementation of the environment management data models, (iii) 3 hours for the implementation of the smart parking management data models, (iv) 6.9 hours for the implementation of the traffic management data models, and (iv) 13.28 hours for the implementation of the air quality management data models.

Regarding implementation tools, SensorThings was able to satisfy two of the defined criteria, offering an SDK[56], as well as a Node.js implementation library of the standard's architecture[57].

[52] www.docs.opengeospatial.org/is/15-078r6/15-078r6.html (Accessed May 30th, 2017)
[53] www.sensorup.atlassian.net/wiki/display/SPS/2015/11/06/%5BTutorial%5D+SensorThings+API+101+How+to+upload+observation+data+to+SensorThings+API (Accessed May 30th, 2017)
[54] As of May 30th, 2017
[55] www.github.com/opengeospatial/sensorthings (Accessed May 30th, 2017)
[56] www.github.com/gost/sensorthings-net-sdk (Accessed May 30th, 2017)
[57] www.npmjs.com/package/sensorthings (Accessed May 30th, 2017)

In the subjective qualitative analysis, SensorThings scored a total of 12 points, distributed as follows:

- **Easiness of Implementation**: 4 points, having a choice of 'Agree', in the 'The data model is easy to implement' Likert Item;
- **Understandability**: 3 points, having a choice of 'Neither Agree or Disagree' in the 'The data model's concepts are easy to understand' Likert Item;
- **Flexibility**: 5 points, having a choice of 'Strongly Agree' in the 'The data model is flexible for changes in the database structure' Likert Item.

### 6.5.2 Quantitative Analysis

After the conversion of Citibrain base model's data into SensorThings' structure, a total of 9 100 model instances were created: 18 *ObservedProperties*, 52 *FeatureOfInterest*, 52 *Location*, 52 *Sensor*, 52 *Thing*, 308 *Datastream*, and 8 566 *Observation*.

For the temporal overhead measurements, the data model scored a total average of 87.64782779 seconds, distributed in the following way:

- **Smart Environment Management:** 5.48 seconds, with a standard deviation of 0.15 seconds;
- **Smart Parking Management:** 18.40 seconds, with a standard deviation of 0.30 seconds;
- **Smart Traffic Management:** 62.31 seconds, with a standard deviation of 0.43 seconds
- **Smart Waste Management:** 1.67 seconds, with a standard deviation of 0.044 seconds.

Regarding the memory overhead measurements, SensorThings scored a total value of 4 396 kilobytes of maximum memory resources used, distributed among the four vertical solutions in the following way:

- **Smart Environment Management:** 638.4 KBs, with a standard deviation of 82.02 KBs;
- **Smart Parking Management:** 1 514 KBs, with a standard deviation of 61.05 KBs;
- **Smart Traffic Management:** 1 299.2 KBs, with a standard deviation of 82.32 KBs;
- **Smart Waste Management:** 944.4 KBs, with a standard deviation of 91.84 KBs;

Concerning the representational overheads, SensorThings achieved the following values for its seven entities:

- **Observed Property**: 40.46 KBs;
- **Thing**: 57.20 KBs;
- **Location**: 20.35 KBs;
- **Feature Of Interest**: 525.11 KBs;
- **Sensor**: 55.06 KBs;
- **Datastream**: 1 366.10KBs;
- **Observation**: 55.06 KBs.

For the Request Overhead measurements, SensorThings scored a total of 10.13 seconds, distributed among the implemented data models in the following way: (i) 2 009 ms for the *Observation* models, (ii) 3 648 ms for the *Datastream* models, (iii) 1 268 ms for the *Feature Of Interest* models, (iv) 1 001 ms for the *Thing* models, (v) 683 ms for the *Location* models, (vi) 876 ms for the *Sensor* models, and (vii) 646 ms for the *Observed Property* models.

After the weighted formula was applied to SensorThings' values, the data model achieved a final score of 0.78.

## 6.6 Conclusions

Upon performing the evaluation of the data models and analyzing the obtained results (observable in Table 4 of the present section), some conclusions can be drawn regarding the level of abstraction's impact on the implementation of the data models:

- Instantiated data models need bigger instantiation periods, but often result in more complete and structured information. Such case was Fiware's. Despite the similar structure to the base model, the data model's implementation time was the third largest. The exception was SensorThings, due to the high granularity of its data model, presented the biggest implementation time of the analyzed models.

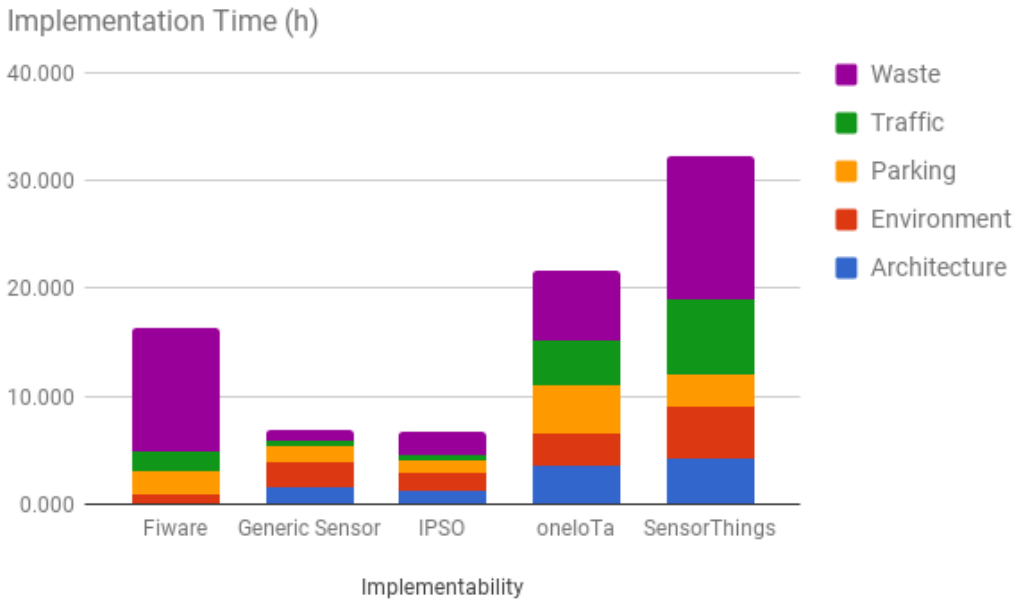



**Figure 26: Comparison of Data Models Implementation Times**

- Abstract models present the advantage of being able to store all current information in the same database tables, as well as additional new information. This translates to the needlessness of applying any changes to or creating new entities in the data model towards the development of new vertical solutions by the smart cities platform.

Regarding the performed analysis, some considerations can also be made.

Overall, Fiware displayed better performance, and ended up being selected for the server implementation of the developed dashboard. Due to the current information structure of Citibrain's data model, the result can be considered positive, and the transition to the new data model a rather simple and straightforward process.

In a close second came IPSO Smart Objects. Despite the simplistic nature of its information structure, the data model revealed itself to be a powerful tool, presenting low implementation times, low overheads, as well as high simplicity and completeness values. Unlike Generic Sensor, IPSO Smart Object's simplistic data model didn't reveal itself as a flaw, with the model being able to represent all Citibrain's entities and relations. The model also achieved the best values in the implementability metric, and scored very similar overhead measurements to Generic Sensor. With the advantages and no disadvantages of the simplicity of its information representation, Smart Objects can be considered a good candidate for Citibrain.

In third and fourth place, with very similar end results came Generic Sensor and SensorThings. While Generic Sensor presented an overly simplistic architecture, not being able to represent all the base model's entities and their relations, the opposite problem can be noted in

SensorThings' structure. Due to the high granularity and redundancy of the information structure, the data model ended up forcing the inclusion of duplicated information in the database, in many cases having the *Sensor* and *Thing* entities representing the same real life entity. Even despite SensorThings' score of the lowest level in the simplicity metric, the model was not able to represent all the base model's entities and their relations, scoring the second lowest value in the *Overlap with co-resident models* metric. However, in spite of its high level of granularity of the information structure and existence of repeated information, SensorThings presented the best results of the qualitative analysis. Generic Sensor also scored one of the lowest support and documentation values, while having no implementation guidelines or tutorials available. On the other hand, due to the data model's simplicity, Generic Sensor scored one of the best results in the overheads measurements. Overall, neither of the two models can be considered a good solution for Citibrain's platform.

Finally, oneIoTa scored the lowest overall results of all the analyzed models. Despite being implemented with a similar architecture to Fiware, the models presented a higher level of granularity, making distinctions between things and locations, which ended up creating some overheads regarding the communication and representation aspects. Another big drawback from the data model is the overheads created in the serialization and implementation processes, due to the need of the inclusion of verbose headers and footers in each model. Due to being a data modelling tool, and the platform's incentive for its users to create and submit new data models, oneIoTa was able to represent all of Citibrain's entities and relationships. Achieving a high score on the subjective analysis, the data model tool proved itself understandable and easy to implement. The data model tool was also the least documented of the studied data models, having no community presence, and only disposing of a repository for the implemented data models, which also serves as examples.

2

**Table 5: Data Model Suitability Evaluation**

| | **Fiware** | **W3C Generic Sensor** | **IPSO Smart Objects** | **oneIoTa** | **OGC SensorThings** |
|---|---|---|---|---|---|
| Temporal Efficiency | 0.84 | 0.81 | 0.66 | 0.73 | 0.96 |
| Memory Efficiency | 0.86 | 0.82 | 0.80 | 0.72 | 0.79 |
| Representational Efficiency | 0.92 | 0.92 | 0.94 | 0.28 | 0.94 |
| Request Efficiency | 0.85 | 0.79 | 0.93 | 0.58 | 0.84 |
| **Quantitative Analysis** | 0.87 | 0.84 | 0.83 | 0.58 | 0.88 |
| Entity Similarity | 1.00 | 0.56 | 1.00 | 1.00 | 0.78 |
| Documentation | 0.95 | 0.43 | 0.58 | 0.29 | 0.57 |
| Simplicity | 0.44 | 0.56 | 0.69 | 0.19 | 0.00 |
| Completeness | 0.44 | 0.56 | 0.69 | 0.19 | 0.00 |
| Implementability | 0.80 | 0.92 | 0.92 | 0.74 | 0.62 |
| Implementation Tools | 1.00 | 0.33 | 1.00 | 0.33 | 0.67 |
| Likert Survey | 0.93 | 1.00 | 0.80 | 0.87 | 0.80 |
| **Qualitative Analysis** | 0.81 | 0.65 | 0.81 | 0.56 | 0.53 |
| **Data Model Quality** | 0.85 | 0.78 | 0.82 | 0.57 | 0.78 |

# Chapter 7

# Conclusions and Future Work

This chapter presents an overall analysis of the developed work, through the confrontation of the results with the proposed goals, the statement of the main contributions of this dissertation and the limitations of the developed framework. The chapter closes with the forecast of future work.

## 7.1 Goal Achievement

The main goal of this project was to perform an analysis on currently available data models for the IoT, with the intent of finding a model that translated into a good solution for Citibrain's platform. In order to achieve such goal, a set of criteria had to be designed in order to characterize what makes a good data model, and, most importantly, what makes a data model the right fit for an IoT system.

In order to answer these questions, a research on data model quality was performed, and a set of criteria was selected. Such criteria were then used to create a data model evaluation framework capable of translating a data model's suitability for a platform into a single value for easy comparison with other models. This framework was then used to test a set of data models' suitability for Citibrain's platform and a solution was found.

All the proposed objectives were met, and the presented solution granted a meaningful contribution to the field of the Internet of Things, since no other data model comparison frameworks directed to the IoT were found during the research stage.

Besides the scientific contributions of this dissertation, the practical component of the project also proved itself very useful for Citibrain. Not only a study was made in order to elect the most suitable data model for the platform, but with the creation of additional databases and

information conversion software, Citibrain's platform is now compliant with the most widely used IoT standards, providing great levels of interoperability to the platform.

## 7.2 Contributions

The biggest scientific contribution of this project was the development of a data model comparison framework. Although the analysis was directed to Citibrain's case, it can be considered generic enough to be applied to other specific cases. The development of criteria arose from the lack of data model evaluation metrics directed to the Internet of Things.

Where some evaluation dimensions and criteria could be found in the research activities, a lack of metrics directed exclusively for the IoT could not be found. Due to the constrained nature of the Internet of Things, some specificities had to been taken into account. Metrics such as the response times of the server containing each data model, or the load introduced in the communication network caused by the representation of each model have a great impact in machine communication, bearing a great importance within the framework's evaluation.

Hence, the devised solution can be considered a good contribution to studies of data model efficiency within the Internet of Things.

## 7.3 Analysis Limitations

The biggest limitation of the presented study is the size of the test group for the developed analysis framework, consisting of only one subject. Based on that fact, some considerations can be made regarding the limitations of the analysis.

- The measured overheads were extremely limited to the implementation method. In an optimal setting, the analysis would be made by averaging the results of the implementations of several developers, in order to achieve a more conformist result.

- The same principle applies, with even more weight, to the subjective part of the qualitative analysis. Given the test audience being composed of just one subject, the obtained results cannot translate into a realistic evaluation of the measured metrics.

For the analysis to have an actual impact, the evaluation should be performed by multiple experts, in order to avoid biases by individual reviewers. As pointed by Moody [22] , other possible way would be to have different experts selected to "apply the metric independently and the results combined into a single rating".

Another limitation falls within the calculation of the request overhead, which only measures the server's response time of the bulk of the registries of a single entity. In the case of abstract models, the information of a single sensor measurement can be stored across several database

tables, and for a correct measurement of the request overhead, the additional queries performed by the server in order to aggregate all the information, should also be taken into account.

## 7.4 Future Work

Although the main goals and objectives of this project were achieved, some considerations can be made in order to improve the framework.

Some future work includes the study and survey of the data models' implementation by a wider test audience. Due to the impact of the conversion software in both the qualitative and the quantitative analysis, metrics such as temporal overhead, memory overhead and implementability, the average of the results of a wider group's implementation is imperative for more precise outcomes.

Also when dealing with a bigger audience, the formula's weights can be modified, in order to give a bigger importance to the subjective metrics, as well as the inclusion of more Likert items in the survey. In order to achieve this, a web platform can be created, enabling IoT developers to share their experiences with the implementation of the data models.

Regarding the performed analysis, some improvements can also be made, such as the inclusion of additional data models in the study, and the inclusion of other metrics specific for Citibrain's platform, such as supported communication protocols. Another improvement for more precise results, would be the development of a complete API for all the models, for a more correct measurement of the request overhead.

# References


[1] M. Batty *et al.*, “Smart cities of the future,” *Eur. Phys. J. Spec. Top.*, vol. 214, no. 1, pp. 481–518, 2012.

[2] S. Consoli *et al.*, “A Smart City Data Model based on Semantics Best Practice and Principles,” *Proc. 24th Int. Conf. World Wide Web Companion*, pp. 1395–1400, 2015.

[3] Citibrain, “Our Solutions,” 2017. [Online]. Available: http://www.citibrain.com/en/solutions/. [Accessed: 26-Jun-2017].

[4] S. Bischof, A. Karapantelakis, A. Sheth, A. Mileo, and P. Barnaghi, “Semantic Modelling of Smart City Data,” *W3C Work. Web Things Enablers Serv. an open Web Devices*, pp. 1–5, 2014.

[5] M. Chisholm, “Ontologies versus Data Models,” 2014. [Online]. Available: http://www.information-management.com/news/news/ontologies-versus-data-models-10025971-1.html. [Accessed: 01-Feb-2017].

[6] M. Koster, “Design Patterns for an Internet Of Things,” 2014. [Online]. Available: http://iot-datamodels.blogspot.pt/2014/05/design-patterns-for-internet-of-things.html. [Accessed: 02-Feb-2017].

[7] T. Berners-lee, J. Hendler, and O. R. A. Lassila, “The Semantic Web will enable machines to,” *Sci. Am.*, no. May, pp. 1–4, 2001.

[8] A. Gyrard, M. Serrano, and G. A. Atemezing, “Semantic web methodologies, best practices and ontology engineering applied to Internet of Things,” *2015 IEEE 2nd World Forum Internet Things*, pp. 412–417, 2015.

[9] European Research Cluster on the Internet of Things, “Internet of Things - IoT Semantic Interoperability: research challeges, best practices, recommendations and next steps,” vol. 1, p. 48, 2015.

[10] S. Consoli, M. Mongiovì, D. Reforgiato, S. Peroni, and A. Gangemi, “The Role of Semantics in Smart Cities Producing Linked Data for Smart Cities : the case of Catania,” vol. 1, pp. 1–5, 2014.

[11] A. J. Jara, A. C. Olivieri, Y. Bocchi, M. Jung, W. Kastner, and A. F. Skarmeta, “Semantic Web of Things: an analysis of the application semantics for the IoT moving towards the IoT convergence,” *Int. J. Web Grid Serv.*, vol. 10, no. 2/3, pp. 244–272, 2014.

[12] C. Bizer, T. Heath, and T. Berners-Lee, “Linked data-the story so far,” *Int. J. Semant. Web Inf. Syst.*, vol. 5, no. 3, pp. 1–22, 2009.

[13] T. Berners-lee, “Linked Data,” 2006. [Online]. Available: https://www.w3.org/DesignIssues/LinkedData.html.

[14] C. Bizer *et al.*, “Linked data on the web,” *WWW2008 Work. Linked Data Web*, pp. 1265–1266, 2008.

[15] World Wide Web Consortium, “Ontologies.” [Online]. Available: https://www.w3.org/standards/semanticweb/ontology. [Accessed: 25-Jan-2017].

[16] B. Chandrasekaran, J. R. Josephson, and V. R. Benjamins, “What aro ontologies, and why do we need them?,” *IEEE Intell. Syst. Their Appl.*, vol. 14, no. 1, pp. 20–26, 1999.

[17] B. S. Institution, “BSI Standards Publication Smart city concept model – Guide to establishing a model for data interoperability,” pp. 1–56, 2014.

[18] A. Gyrard, C. Bonnet, K. Boudaoud, and M. Serrano, “LOV4IoT: A second life for ontology-based domain knowledge to build Semantic Web of Things applications,” *IEEE 4th Int. Conf. Futur. Internet Things Cloud*, 2016.

[19] A. Gyrard and M. Serrano, “Connected Smart Cities : Interoperability with SEG 3 . 0 for the Internet of Things,” no. 2, pp. 1–7, 2016.

[20] R. Parundekar, C. A. Knoblock, and L. Ambite, “Linking and Building Ontologies of Linked Data | ISWC 2010.”

[21] R. Maier, “Evaluation of Data Modeling,” 1995.

[22] D. L. Moody and G. G. G. Shanks, “What Makes a Good Data Model? Evaluating the Quality of Entity Relationship Models,” *Proc. the13th Int. Conf. Entity-relationsh. Approach*, pp. 94–110, 1994.

[23] W. C. McGee, “On user criteria for data model evaluation,” *ACM Trans. Database Syst.*, vol. 1, no. 4, pp. 370–387, 1976.

[24] A. Joshi, S. Kale, S. Chandel, and D. Pal, “Likert Scale: Explored and Explained,” *Br. J. Appl. Sci. Technol.*, vol. 7, no. 4, pp. 396–403, 2015.

[25] H. N. J. Boone and D. A. Boone, “Analyzing Likert data,” *J. Ext.*, vol. 50, no. 2, p. 30, 2012.

[26] A. Kostoulas, “Designing Better Questionaires: Using Scales,” *Designing Better Questionaires: Using Scales*, 2014. [Online]. Available: https://achilleaskostoulas.com/2014/02/03/designing-better-questionnaires-using-likert-scales/. [Accessed: 19-Apr-2017].

[27] A. K. B, Z. Jan, A. Zappa, and M. Serrano, “Interoperability and Open-Source Solutions for the Internet of Things,” vol. 10218, pp. 20–35, 2017.

[28] FIWARE, “About us,” 2016. [Online]. Available: https://www.fiware.org/about-us/. [Accessed: 06-Jan-2017].

[29] OGC, S. Liang, A. C. Huang, T. Khalafbeigi, and K. Kim, “OGC SensorThings API Candidate Standard Part 1: Sensing,” pp. 1–88, 2015.

[30] CitySDK, “Issue reporting / Open311 API.” [Online]. Available: https://www.citysdk.eu/citysdk-toolkit/using-the-apis/open311-api/. [Accessed: 12-Jan-2017].

[31] CitySDK, “Tourism API.” [Online]. Available: https://www.citysdk.eu/citysdk-toolkit/using-the-apis/tourism-api/. [Accessed: 12-Jan-2017].

[32] CitySDK, “Linked Data API.” [Online]. Available: https://www.citysdk.eu/citysdk-toolkit/using-the-apis/linked-data-platform. [Accessed: 12-Jan-2017].

[33] M. Pizzo, R. Handl, and M. Zurmuehl, “OData Version 4.0 Part 1: Protocol,” *OASIS Stand.*, no. February, pp. 1–69, 2014.

[34] World Wide Web Consortium, “Generic Sensor API,” 2016. [Online]. Available: https://www.w3.org/TR/2016/WD-generic-sensor-20160830/. [Accessed: 28-Jan-2017].

[35] J. Jimenez, M. Koster, and H. Tschofenig, “IPSO Smart Objects,” 2016.

[36] Open Connectivity Foundation, “oneIoTa User Guide,” 2016.

[37] IPSO Alliance, “IPSO SmartObject Guideline - Smart Objects Starter Pack1.0,” pp. 1–39, 2014.

[38] FIWARE, “Publish/Subscribe Context Broker - Orion Context Broker,” 2015. [Online]. Available: https://catalogue.fiware.org/enablers/publishsubscribe-context-broker-orion-context-broker. [Accessed: 06-Jan-2017].

[39] Alliance Open Mobile, “Lightweight Machine to Machine Requirements,” pp. 1–112, 2012.

[40] Sap Hana, “OData Architecture.” [Online]. Available: http://saphanatutorial.com/odata-architecture/. [Accessed: 16-Jan-2017].

[41] Open Geospatial Consortium, “SensorThings Data Model,” 2015. [Online]. Available: http://ogc-iot.github.io/ogc-iot-api/datamodel.html. [Accessed: 23-Jan-2017].

# Annex A

# Data Model Evaluation Measurements

This annex presents the extensive measurements for the values calculated and presented in Chapter 6. Tables are presented for both the quantitative and qualitative analysis and expand all approached metrics.

- **Qualitative Analysis**
  - **Overlap with co-resident models**

| Overlap | Citibrain | Fiware | Generic Sensor | IPSO | oneIoTa | SensorThings |
|---|---|---|---|---|---|---|
| **Generic** | Asset | Device | Sensor + SensorType | Object - Sensor | Sensor | Sensor + Location + Datastream |
| **Environment** | Environment Event | AirQualityObserved | Reading | Object (Sensor Value + Units) | AirQuality + TimePeriod | Observation |
| **Parking** | Parking Area | OnStreetParking | --- | Composite Object | Area | --- |
| | Parking Spot | ParkingSpot | --- | Composite Object | Parking Spot | Thing + Location + ObsProp + FOI |
| | Park Event | ParkingEvent | Reading | Object (Sensor Value + Units) | Occupancy + TimePeriod | Observation + ObsProp |
| **Traffic** | Traffic Event | TrafficFlowObserved | Reading | Object (Sensor Value + Units) | Activity Count + TimePeriod | Observation + ObsProp |
| **Waste** | Waste Area | WasteContainerIsle | ---- | Composite Object | Area | --- |
| | Waste Container | WasteContainer | --- | Composite Object | Waste Container | Thing + ObsProp + FOI |
| | Waste Event | WasteEvent | Reading | Object (Sensor Value + Units) | Fill level + temperature + turned over + TimePeriod | Observation + ObsProp |
| **Total** | 9 | 9 | 5 | 9 | 9 | 7 |

- **Documentation Analysis**

| Documentation | Fiware | Generic Sensor | IPSO | oneIoTa | SensorThings |
|---|---|---|---|---|---|
| **Specification ?** | 1 | 1 | 1 | 0 | 1 |
| **Guidelines?** | 1 | 0 | 1 | 0 | 1 |
| **Tutorial?** | 1 | 0 | 1 | 0 | 1 |
| **Repository?** | 1 | 1 | 1 | 1 | 1 |
| Community? | 172 | N/A | 0 | 0 | 0 |
| Unanswered in Community | 4 | N/A | 0 | 0 | 0 |
| Community Responsiveness | 0.977 | N/A | 0.000 | 0.000 | 0.000 |
| **Community Normalized** | 0.977 | 0.000 | 0.000 | 0.000 | 0.000 |
| StackOverflow | 727 | 4 | 33 | 0 | 9 |
| Unanswered in StackOverflow | 218 | 2 | 6 | 0 | 3 |
| StackOverflow Responsiveness | 0.700 | 0.500 | 0.818 | 0.000 | 0.667 |
| **StackOverflow Normalized** | 0.700 | 0.003 | 0.037 | 0.000 | 0.008 |
| **Examples?** | 1 | 1 | 0 | 1 | 0 |
| **Total** | 6.677 | 3.003 | 4.037 | 2.000 | 4.008 |
| **Total Normalized** | 0.954 | 0.429 | 0.577 | 0.286 | 0.573 |

- **Simplicity**

| Simplicity | Fiware | Generic Sensor | IPSO | oneIoTa | SensorThings |
|---|---|---|---|---|---|
| **Entities** | 5 | 4 | 3 | 6 | 8 |
| **Relations** | 4 | 3 | 2 | 7 | 8 |
| **Total (E+R)** | 9 | 7 | 5 | 13 | 16 |
| **Total Nomalized** | 0.563 | 0.438 | 0.313 | 0.813 | 1.000 |
| **Reversed** | 0.438 | 0.563 | 0.688 | 0.188 | 0.000 |

- **Completeness**

| Completeness | Citibrain | Fiware | Generic Sensor | IPSO | oneIoTa | SensorThings |
|---|---|---|---|---|---|---|
| **Entities** | 4 | 5 | 4 | 3 | 6 | 8 |
| **Relations** | 4 | 4 | 3 | 2 | 7 | 8 |
| **Total (E+R)** | 8 | 9 | 7 | 5 | 13 | 16 |
| **Nomalized** | 1.000 | 1.125 | 0.875 | 0.625 | 1.625 | 1.000 |
| **Reversed** | 0.500 | 0.438 | 0.563 | 0.688 | 0.188 | 0.500 |

- **Implementability**

| Implementability | Fiware | Generic Sensor | IPSO | oneIoTa | SensorThings |
|---|---|---|---|---|---|
| **Architecture** | N/A | 1.500 | 1.180 | 3.500 | 4.300 |
| **Environment** | 0.850 | 2.400 | 1.730 | 3.000 | 4.720 |
| **Parking** | 2.180 | 1.470 | 1.170 | 4.580 | 3.000 |
| **Traffic** | 1.850 | 0.570 | 0.550 | 4.030 | 6.900 |
| **Waste** | 11.500 | 1.000 | 2.040 | 6.470 | 13.280 |
| **Total** | 16.380 | 6.940 | 6.670 | 21.580 | 32.200 |
| **Normalized** | 0.804 | 0.917 | 0.920 | 0.742 | 0.616 |

- **Implementation Tools**

| Implementation Tools | Fiware | GenericSensor | IPSO | oneIoTa | SensorThings |
|---|---|---|---|---|---|
| **SDK** | 1 | 0 | 1 | 1 | 1 |
| **Libraries** | 1 | 0 | 1 | 0 | 1 |
| **Validation Tools** | 1 | 1 | 1 | 0 | 0 |
| **Total** | 3 | 1 | 3 | 1 | 2 |
| **Normalized** | 1 | 0.333 | 1 | 0.333 | 0.667 |

- **Subjective Analysis**

| Likert Analysis | Fiware | GenericSensor | IPSO | oneIoTa | SensorThings |
|---|---|---|---|---|---|
| **Easiness of Implementation** | 5 | 5 | 3 | 4 | 4 |
| **Understandability** | 5 | 5 | 4 | 5 | 3 |
| **Flexibility** | 4 | 5 | 5 | 4 | 5 |
| **Total** | 14 | 15 | 12 | 13 | 12 |
| **Total (Normalized)** | 0,933 | 1 | 0,800 | 0,867 | 0,800 |

# • Quantitative Analysis

- **Temporal Overhead Measurements**

| Environment | FIWARE | GenericSensor | IPSO | oneIoTa | SensorThings |
|---|---|---|---|---|---|
| **1** | 0.277 | 2.487 | 2.775 | 0.635 | 5.502 |
| **2** | 0.280 | 2.524 | 2.787 | 0.616 | 5.909 |
| **3** | 0.272 | 2.518 | 2.628 | 0.631 | 5.405 |
| **4** | 0.283 | 2.526 | 2.678 | 0.661 | 5.377 |
| **5** | 0.272 | 2.593 | 2.665 | 0.640 | 5.491 |
| **6** | 0.273 | 2.585 | 2.876 | 0.684 | 5.442 |
| **7** | 0.271 | 2.552 | 2.767 | 0.671 | 5.385 |
| **8** | 0.286 | 2.629 | 2.703 | 0.641 | 5.423 |
| **9** | 0.265 | 2.576 | 2.683 | 0.638 | 5.430 |
| **10** | 0.281 | 2.573 | 2.859 | 0.661 | 5.443 |
| **Mean (s)** | 0.276 | 2.556 | 2.742 | 0.648 | 5.481 |
| **STDEV (s)** | 0.006 | 0.041 | 0.080 | 0.020 | 0.148 |

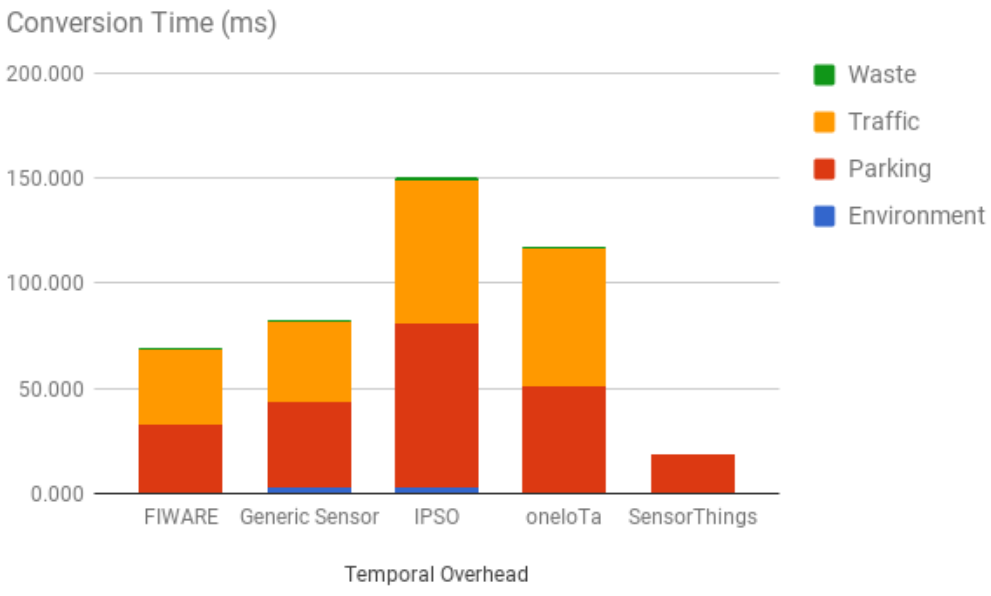
| Parking | FIWARE | GenericSensor | IPSO | oneIoTa | SensorThings |
|---|---|---|---|---|---|
| **1** | 32.854 | 40.968 | 77.911 | 50.604 | 18.189 |
| **2** | 30.652 | 36.477 | 74.827 | 50.300 | 17.882 |
| **3** | 30.303 | 36.025 | 74.789 | 50.907 | 18.436 |
| **4** | 29.920 | 36.191 | 74.555 | 49.987 | 18.008 |
| **5** | 30.103 | 35.936 | 74.379 | 50.025 | 18.221 |
| **6** | 30.252 | 36.829 | 75.362 | 50.188 | 18.834 |
| **7** | 30.705 | 37.052 | 75.115 | 50.208 | 18.251 |
| **8** | 30.304 | 36.041 | 74.157 | 50.720 | 18.720 |
| **9** | 30.638 | 36.239 | 75.902 | 51.225 | 18.584 |
| **10** | 31.123 | 36.639 | 74.124 | 50.804 | 18.398 |
| **Mean (s)** | 30.685 | 36.840 | 75.112 | 50.497 | 18.352 |
| **STDEV (s)** | 0.837 | 1.497 | 1.127 | 0.416 | 0.302 |

| Traffic | FIWARE | GenericSensor | IPSO | oneIoTa | SensorThings |
|---|---|---|---|---|---|
| **1** | 36.891 | 40.631 | 68.395 | 66.707 | 61.759 |
| **2** | 35.167 | 37.605 | 67.255 | 65.210 | 62.420 |
| **3** | 35.555 | 38.298 | 68.604 | 64.529 | 61.758 |
| **4** | 35.066 | 37.248 | 67.736 | 64.596 | 61.812 |
| **5** | 35.092 | 37.560 | 67.783 | 65.097 | 62.310 |
| **6** | 35.775 | 38.211 | 68.217 | 65.264 | 63.162 |
| **7** | 35.528 | 37.720 | 68.727 | 64.920 | 62.541 |
| **8** | 35.554 | 37.869 | 70.482 | 65.266 | 62.488 |
| **9** | 35.561 | 38.271 | 68.203 | 65.392 | 62.693 |
| **10** | 36.145 | 37.968 | 68.932 | 65.450 | 62.169 |
| **Mean (s)** | 35.633 | 38.138 | 68.433 | 65.243 | 62.311 |
| **STDEV (s)** | 0.522 | 0.892 | 0.834 | 0.571 | 0.430 |

| Waste | FIWARE | GenericSensor | IPSO | oneIoTa | SensorThings |
|---|---|---|---|---|---|
| **1** | 0.667 | 0.706 | 1.325 | 1.049 | 1.642 |
| **2** | 0.508 | 0.661 | 1.357 | 1.048 | 1.648 |
| **3** | 0.516 | 0.684 | 1.425 | 1.009 | 1.671 |
| **4** | 0.523 | 0.747 | 1.335 | 1.002 | 1.677 |
| **5** | 0.518 | 0.715 | 1.361 | 1.018 | 1.587 |
| **6** | 0.511 | 0.652 | 1.333 | 0.999 | 1.689 |
| **7** | 0.510 | 0.674 | 1.399 | 0.988 | 1.693 |
| **8** | 0.557 | 0.665 | 1.349 | 1.044 | 1.759 |
| **9** | 0.540 | 0.690 | 1.377 | 1.015 | 1.619 |
| **10** | 0.550 | 0.647 | 1.358 | 1.033 | 1.683 |
| **Mean (s)** | 0.540 | 0.684 | 1.362 | 1.021 | 1.667 |
| **STDEV** | 0.046 | 0.030 | 0.030 | 0.021 | 0.044 |

| Total | FIWARE | GenericSensor | IPSO | oneIoTa | SensorThings |
|---|---|---|---|---|---|
| **Environment** | 0.276 | 2.556 | 2.742 | 0.648 | 0.148 |
| **Parking** | 32.854 | 40.968 | 77.911 | 50.604 | 18.189 |
| **Traffic** | 35.633 | 38.138 | 68.433 | 65.243 | 0.430 |
| **Waste** | 0.540 | 0.684 | 1.362 | 1.021 | 0.044 |
| **Total** | 69.303 | 82.346 | 150.448 | 117.516 | 18.810 |

- **Graphical Representation of Temporal Overhead**



- **Memory Overhead**

| Environment | FIWARE | GenericSensor | IPSO | oneIoTa | SensorThings |
|---|---|---|---|---|---|
| **1** | 288 | 824 | 572 | 632 | 436 |
| **2** | 280 | 1004 | 636 | 568 | 696 |
| **3** | 276 | 972 | 524 | 696 | 664 |
| **4** | 284 | 864 | 524 | 636 | 696 |
| **5** | 276 | 1036 | 632 | 696 | 632 |
| **6** | 284 | 1036 | 664 | 696 | 696 |
| **7** | 284 | 908 | 632 | 636 | 636 |
| **8** | 284 | 864 | 696 | 636 | 696 |
| **9** | 276 | 972 | 520 | 664 | 664 |
| **10** | 284 | 912 | 636 | 572 | 568 |
| **Mean (kB)** | 281.600 | 939.200 | 603.600 | 643.200 | 638.400 |
| **STDEV (kB)** | 4.300 | 75.561 | 63.734 | 46.906 | 82.023 |

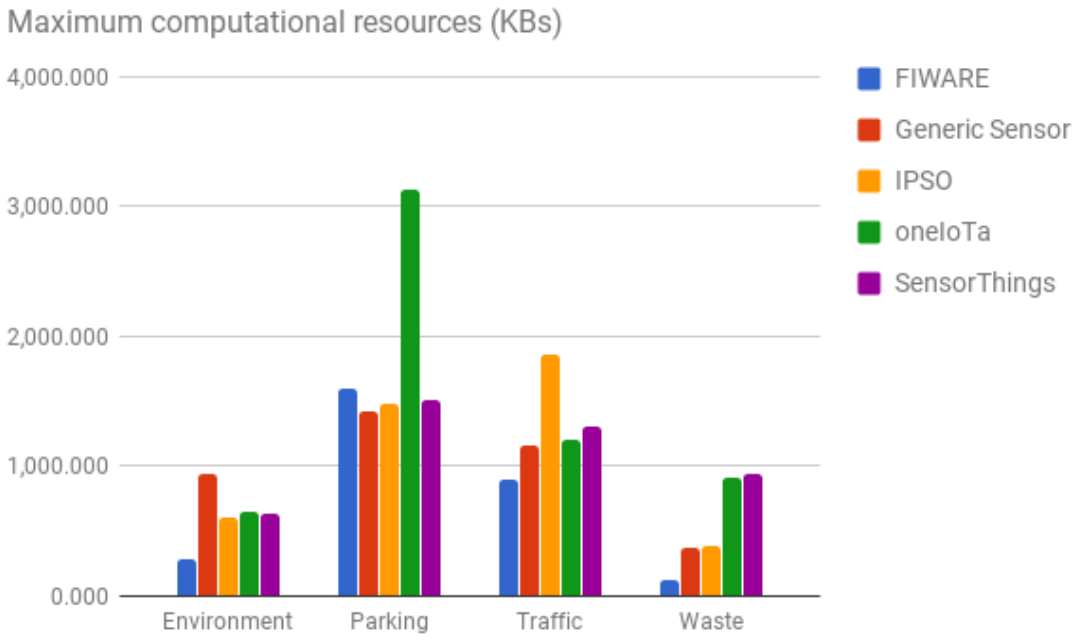
| Parking | FIWARE | GenericSensor | IPSO | oneIoTa | SensorThings |
|---|---|---|---|---|---|
| **1** | 1632 | 1664 | 1584 | 3256 | 1580 |
| **2** | 1536 | 1340 | 1424 | 3156 | 1392 |
| **3** | 1604 | 1404 | 1368 | 3096 | 1564 |
| **4** | 1604 | 1276 | 1428 | 3156 | 1504 |
| **5** | 1532 | 1408 | 1488 | 3124 | 1564 |
| **6** | 1532 | 1404 | 1560 | 3036 | 1500 |
| **7** | 1656 | 1464 | 1488 | 3156 | 1436 |
| **8** | 1660 | 1404 | 1492 | 3156 | 1532 |
| **9** | 1600 | 1404 | 1424 | 3096 | 1564 |
| **10** | 1660 | 1404 | 1556 | 3028 | 1504 |
| **Mean (kB)** | 1,601.60 | 1,417.20 | 1,481.20 | 3,126.00 | 1,514.00 |
| **STDEV (kB)** | 52.40 | 99.97 | 70.53 | 66.67 | 61.05 |

| Traffic | FIWARE | GenericSensor | IPSO | oneIoTa | SensorThings |
|---|---|---|---|---|---|
| **1** | 984 | 1292 | 1932 | 1212 | 1128 |
| **2** | 912 | 1148 | 1936 | 1148 | 1260 |
| **3** | 888 | 1080 | 1808 | 1144 | 1328 |
| **4** | 888 | 1144 | 1808 | 1148 | 1388 |
| **5** | 888 | 1144 | 1936 | 1464 | 1388 |
| **6** | 820 | 1148 | 1808 | 1208 | 1260 |
| **7** | 944 | 1152 | 1932 | 1208 | 1324 |
| **8** | 820 | 1148 | 1932 | 1208 | 1392 |
| **9** | 944 | 1188 | 1764 | 1208 | 1260 |
| **10** | 944 | 1144 | 1808 | 1144 | 1264 |
| **Mean (kB)** | 903.2 | 1158.8 | 1866.4 | 1209.2 | 1299.2 |
| **STDEV (kB)** | 53.89 | 53.60 | 72.05 | 94.83 | 82.32 |

| Waste | FIWARE | GenericSensor | IPSO | oneIoTa | SensorThings |
|---|---|---|---|---|---|
| **1** | 128 | 304 | 384 | 900 | 712 |
| **2** | 124 | 384 | 384 | 820 | 968 |
| **3** | 124 | 384 | 384 | 984 | 972 |
| **4** | 124 | 384 | 384 | 984 | 908 |
| **5** | 124 | 384 | 384 | 928 | 972 |
| **6** | 124 | 384 | 384 | 952 | 1032 |
| **7** | 124 | 384 | 384 | 984 | 972 |
| **8** | 124 | 384 | 384 | 820 | 904 |
| **9** | 124 | 384 | 384 | 928 | 972 |
| **10** | 124 | 384 | 384 | 860 | 1032 |
| **Mean (kB)** | 124.4 | 376 | 384 | 916 | 944.4 |
| **STDEV (kB)** | 1.26 | 25.30 | 0.00 | 64.36 | 91.84 |

| Total | FIWARE | GenericSensor | IPSO | oneIoTa | SensorThings |
|---|---|---|---|---|---|
| **Environment** | 281.600 | 939.200 | 603.600 | 643.200 | 638.400 |
| **Parking** | 1,601.60 | 1,417.20 | 1,481.20 | 3,126.00 | 1,514.00 |
| **Traffic** | 903.2 | 1158.8 | 1866.4 | 1209.2 | 1299.2 |
| **Waste** | 124.4 | 376 | 384 | 916 | 944.4 |
| **Total** | 2,910.80 | 3,891.20 | 4,335.20 | 5,894.40 | 4,396.00 |

- **Graphical Representation of Memory Overhead**



- **Representational and Request Overheads**

| FIWARE | Size (B) | Time (ms) |
|---|---|---|
| **device** | 4512 | 1873 |
| **air_quality_observed** | 4612 | 842 |
| **on_street_parking** | 1481 | 777 |
| **parking_spot** | 2956 | 628 |
| **traffic_flow_observed** | 2665617 | 3314 |
| **waste_container** | 8121 | 757 |
| **waste_container_isle** | 8121 | 627 |
| **waste_container_model** | 645 | 628 |
| **Total** | 2696065 | 9446 |

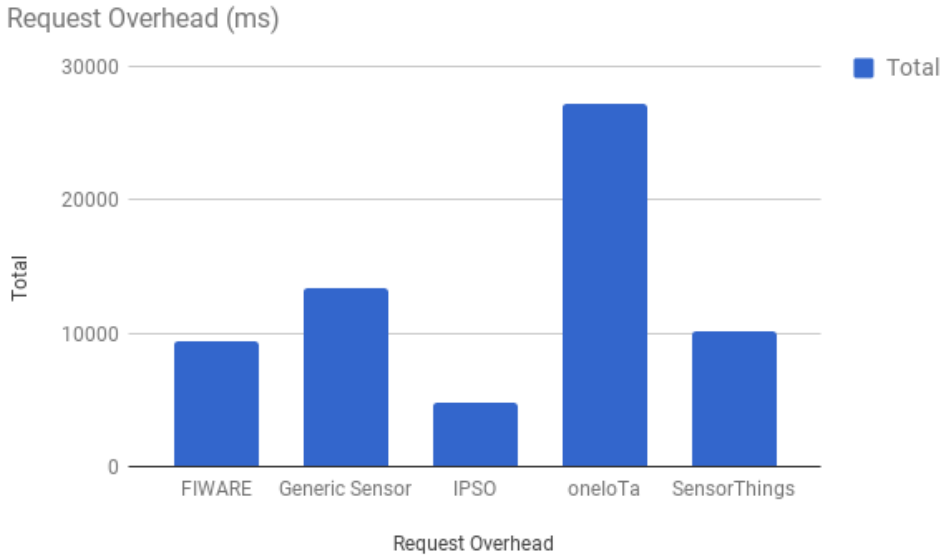
| GenericSensor | Size (B) | Time (ms) |
|---|---|---|
| **SensorType** | 11930 | 1263 |
| **Sensor** | 405590 | 1735 |
| **Reading** | 2252818 | 10365 |
| **Total** | 2670338 | 13363 |

| IPSO | Size (B) | Time (ms) |
|---|---|---|
| **Composite** | 21524 | 1813 |
| **Object** | 507808 | 1469 |
| **Resource** | 1459497 | 1515 |
| **Total** | 1988829 | 4797 |

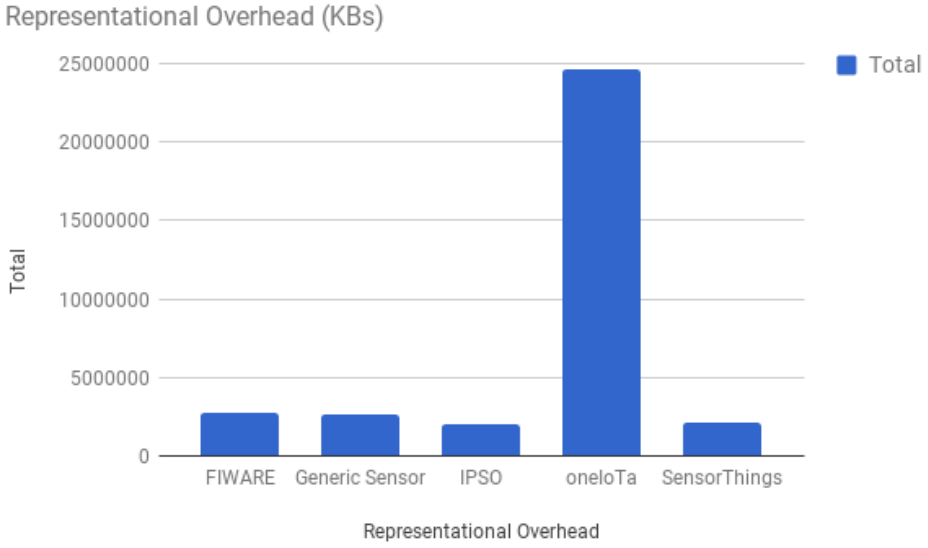
| oneIoTa | Size (B) | Time (ms) |
|---|---|---|
| **time_period** | 12368191 | 10114 |
| **turned_over** | 16066 | 654 |
| **waste_container** | 22899 | 693 |
| **temperature** | 16044 | 633 |
| **activity_count** | 7364832 | 4611 |
| **air_quality** | 172059 | 3947 |
| **fill_level** | 16165 | 695 |
| **occupancy** | 4389146 | 1631 |
| **parking_spot** | 140785 | 859 |
| **sensor** | 69906 | 924 |
| **geolocation** | 45971 | 948 |
| **area_poly (area)** | 4725 | 728 |
| **area (area_small)** | 4368 | 787 |
| **Total** | 24631157 | 27224 |

| SensorThings | Size (B) | Time (ms) |
|---|---|---|
| **observation** | 55055 | 2009 |
| **datastream** | 1366114 | 3648 |
| **feature_of_interest** | 525106 | 1268 |
| **thing** | 57203 | 1001 |
| **location** | 20349 | 683 |
| **sensor** | 55055 | 876 |
| **observed_property** | 40457 | 646 |
| **Sum (KB, ms)** | 2119339 | 10131 |

- **Graphical Representation of Request Overheads**




- **Graphical Representation of Representational Overheads**



# Annex B

# A Data Model Evaluation Framework for the IoT

This annex presents a scientific paper summing up the development process of the framework, the state of the art of data model evaluation, as well as the achieved results and conclusions. The paper is to be submitted to IEEE's World Forum on the Internet of Things[58], to be held between February 5th and 8th, 2018.

[58] http://wfiot2018.ieee-wf-iot.org/

# A Data Model Evaluation Framework for the Internet of Things


## Abstract

In a time where societies are committed to moving towards more sustainable solutions in what concerns their infrastructures, the Internet of Things is empowering the smart city sector, playing an increasingly important role in areas such as environment quality, mobility, security and public health. However, due to the diversity of the sensed information and the formats it is conveyed in, coupled with the vastness of existing standards and protocols, there are great efforts and time spent by the developers in order to manually adapt their applications to these incongruous formats and to ensure compatibility across systems. One solution to overpass this problem is the adoption of data models as means of achieving both syntactic and semantic interoperability. This paper proposes a framework to compare and rank existing data models for the Internet of Things, within the smart city sector. A literature review on already devised methodologies for the evaluation of data models is performed, and, using the deemed most relevant criteria, along with IoT-specific metrics, a framework is devised, capable of translating a data model's suitability for a platform into a single value, for easy comparison between existing alternatives. The framework is then applied to a smart cities platform, Citibrain, in a practical use case.



**Keywords**: Internet of Things, Data Interoperability, Smart Cities, Data Model, Framework

# 1. Introduction

Created in the beginning of the 21st century, the concept of smart city, "a city in which information and communication technologies (ICT) is merged with traditional infrastructures, coordinated and integrated using new digital technologies" [1], has been in constant refinement. With recent improvements in ICT, resulting in ubiquitous Internet access and lower prices in mobile devices and sensors, the Internet of Things (IoT) became an ever-increasing market, tightly linked with the empowerment of the smart city sector [2].

Smart city data is obtained through sensory devices that measure different types of observations, such as light, movement or temperature. These sensors provide data of different and changing quality, and, with this data often being continuous, the feed will result in different validity and availability over time, translating to highly dynamic data streams [3].

Smart city data is also vast and heterogeneous in nature: being obtained from many different sectors (e.g.: traffic information, home automation, environment quality); being of different nature even within the same sector, i.e. to measure environment quality one would have to access metrics such as humidity, temperature and gas concentration levels in the atmosphere; as well as having different update rates.

Bischof et al. [3] identify three different types of update rate related to the data obtained from a city:

- **Static Data**: Data never or rarely updated (e.g.: points of interest);
- **Semi-Dynamic Data**: Data updated periodically (e.g.: cultural events);
- **Dynamic Data**: Data continuously updated (e.g.: traffic flow).



**Table 1: Information potentially retrieved from cities** [3]

| Data Category | Owner (Data Publisher) | Data Description | Sampling |
|---|---|---|---|
| **Transport** | Traffic Authority | Maps of Cities (Roads, Street Names, POIs, etc) | Static |
| | Municipality | Public Transport Schedules | Semi-Dynamic |
| | Traffic Authority | Transport Authority Updates (Roadwork, traffic status, etc) | Dynamic |
| **Air Quality** | Environment Agency | Particle Concentration | Dynamic |

| | | | |
|---|---|---|---|
| **Traffic** | Traffic Authority | Number of vehicles passing between two points, speed | Dynamic |
| **City Events** | Cultural Groups | Entertainment (movie/theater plays) | Semi-Dynamic |
| **Municipal Services** | Municipality | Library Data | Dynamic |
| | Private Company | Parking Meters | Dynamic |
| **Citizen Data** | Private Individuals | Social Media Information: Tweets, Status updates and blog posts, popular places ("check-ins") | Semi-Dynamic |
| | | Household Energy Consumption | Semi-Dynamic |
| **Health Data** | Private and Public | Relevant information about potential or confirmed sources of health threats | Dynamic |

After being collected, this data is usually transferred over several systems, being integrated into lower-level (as is the case for complex sensors) or high-level (end-user) applications, as well as being made available via query or publish/subscribe services [3]. All these processes create additional levels of heterogeneity. For a better understanding of this smart city data flux, the following example will illustrate a concrete use case for the measurement of air quality in a city zone. An air quality sensor is usually a complex sensor, composed of several smaller sensors, responsible for measuring metrics such as temperature, gas concentration, luminance, precipitation, and noise levels. The complex air quality sensor is then responsible for the aggregation of the information provided by the small sensors, and sending it to an air quality management system, which will analyze the data and adapt it into the platform's data models. Besides the analysis by the management system, the processed, structured information is usually made available by the platform to websites, mobile applications and in the aforementioned publish/subscribe services, in the form of an API server. All these operations manipulate the sensed data and add different types of metadata (headers, representational terms) in order to better adapt the information to the systems, but, at the same time, creating barriers to the interoperability between the system and outer systems.

Extracting meaningful information from a smart city is a somewhat complex task, given the co-existence of the aforementioned levels of heterogeneity created by the information flux and the overall volume of the obtained data. Then, research on the processes of data interoperability becomes of primary interest [4].

As a result of the vastness and diversity of data, combined with the ever-increasing size of the IoT market, the heterogeneity problem goes even further. As the IoT, and specifically the smart city market continues to grow and get increasingly more attention by both the industry and the research community, IoT-powered applications are being rapidly developed in a great number of domains, such as environment, mobility, and healthcare. In order to keep track with this growth, companies and organizations have been betting in the development of new applications for the IoT, often resulting in the emergence of heterogeneous architectures, standards, middleware platforms and applications [5].

Kazmi et al. [5] identified three areas of the IoT where heterogeneity is introduced through the aforesaid developments, and highlighted some efforts currently being made by organizations:

- **Architectures and Standards**: Organizations and consortiums such as Fiware[1], Open Mobile Alliance[2], Eclipse IoT[3], the World Wide Web Consortium (W3C)[4], IPSO Alliance[5], the European Telecommunications Standards Institute (ETSI)[6], and the Internet Engineering Task Force (IETF)[7] have been developing new standards towards the harmonization of device communication;

- **Middleware Platforms**: In order to ease the collection of data from homogeneous and heterogeneous sources, a number of middleware platforms have been developed, such as Hayo[8], Cisco Flare[9] and SensorWare[10]. These platforms provide a plethora of functionalities, such as the collection of data from physical sensors, the transformation of such data into new structures and the enablement of interfaces for applications, and are usually also combined with other ICT technologies such as cloud computing. One of the bigger goals of these platforms is the encouragement of end-users to connect their IoT devices to the cloud infrastructure and the enablement of APIs for the attainment of sensory information and the development of applications;

- **Applications**: Recent times have witnessed the development of many IoT applications, either for research purposes or for commercial ones. These applications are often

[1] www.fiware.org
[2] openmobilealliance.org
[3] iot.eclipse.org
[4] www.w3.org
[5] www.ipso-alliance.org
[6] www.etsi.org
[7] www.ietf.org
[8] www.hayo.io
[9] developer.cisco.com/site/flare
[10] sensorware.sourceforge.net

developed for very specific domains of a smart city, such as smart parking or waste management solutions, and very frequently make use of numerous IoT devices.

This paper focuses on the study of data models for the IoT as a solution for the standardization of the information provided by different sensors measuring data of different nature. Such standardization serves as a means to achieve the much needed data interoperability among smart city IoT platforms. More specifically, this paper focuses on the case of Citibrain, a technology consortium that provides Internet of Things solutions that use the same stack and technological architecture (from now on referred to as vertical solutions) in the domains of mobility and environment quality, with the purpose of converging as a platform for smart cities. Smart Air Quality Management, Smart Waste Management, Smart Parking Management and Smart Traffic Management are the four vertical solutions currently offered by Citibrain. The main goal of the performed study is to find a data model for the Internet of Things that fits Citibrain's solutions.

The rest of the paper is structured as follows: Section 2 presents a literature review on data interoperability in IoT systems, the semantic web approach, and previous studies on the evaluation of data models. Section 3 presents an analysis of discovered data models for the IoT, and section 4 contains the evaluation of the selected models. Section 5 displays the results of the performed evaluation. Finally, Section 6 presents the conclusions and future work.



## 2. The IoT and Smart City Semantics

This section reviews previous work in the field of the IoT, related to its smart city sector and quality evaluation of data models, and includes: (i) a study on the field of data interoperability (ii) a brief description of the semantic web and two of its technologies, Linked Data and Ontologies, and how they apply to the IoT, and (iii) an analysis of data model evaluation techniques and the quantification methods of qualitative criteria.

### 2.1. Data Interoperability in IoT Systems

In their 2015 paper, Serrano et al. [6] identify four main challenges regarding data interoperability in IoT systems:

- **Technical Interoperability** – concerning heterogeneous software and hardware, such as communication protocols;

- **Syntactical Interoperability** – concerning both different data formats and different software for the development of ontologies and semantic datasets;
- **Semantic Interoperability** – ontology heterogeneity, differences in terms used to describe data, as well as the meaning of the exchanged data according to the context;
- **Organizational Interoperability** – concerning heterogeneity of different infrastructures.

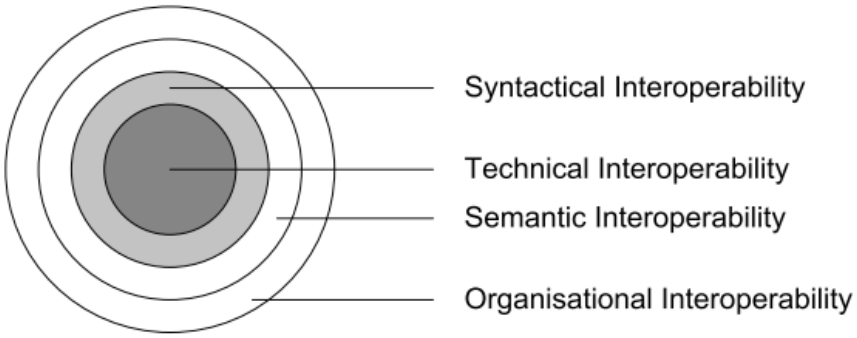



**Figure 1: The four challenges of data interoperability** [7]

According to Consoli et al. [4], the large and heterogeneous data sources of smart cities increase the problem, since different semantic perspectives need to be taken into account, in order to deal with knowledge source conceptualizations. In the same paper, answers to these challenges are pointed out, through the employment of semantic web standards and practices. The authors consider that syntactical interoperability, which regards the used data formats, can be achieved by the adoption of universal knowledge representation languages, such as Resource Description Frameworks (RDF), Web Ontology Languages (OWL) or Linked Open Data (LOD). Besides, it is proposed that semantic interoperability, which regards the meaning of the exchanged data and the different terms used to describe it, may be achieved through the use of a uniform data representation, as well as the formalization of all the concepts into a holistic data model.



## 2.2. The Semantic Web Approach

In recent years, the Internet of Things or, more precisely, its connectivity service - the Web of Things, started integrating semantic web technologies [8]. The Semantic Web, an extension of the Web through standards provided by the World Wide Web Consortium aims at, according to Berners-Lee et al [9], providing structure and meaning to the web pages' content, thus creating

an "environment where software agents roaming from page to page can readily carry out sophisticated tasks for users". In the same paper, the authors describe the Semantic Web as an extension of the current Web, where information and data can be processed automatically, with the contents also being destined to be read and comprehended by machines, instead of being only human-readable as is the case with the current Web. By adding logic to the Web, and giving computers access to structured collections of data and sets of inferences rules, computers become able to conduct automated reasoning, to make inferences and to choose courses of action and answer questions, making use of technologies such as universal knowledge representation languages.

By merging the notions of Semantic Web and the Web of Things, a new concept was created, the Semantic Web of Things (SWoT). The SWoT envisions the integration into the physical world of information which is simultaneously rich in semantic terms, and easily accessible; it also aims at connecting smart objects and digital entities. According to Jara et al.[8], the SWoT purpose is the enablement of "knowledge-based systems to achieve high degrees of autonomous capability for information storage, management, and discovery, therefore providing transparent access to information sources", in a reality dominated by constrained devices, with low memory capacity, little processing capabilities and low throughput wireless links [9].

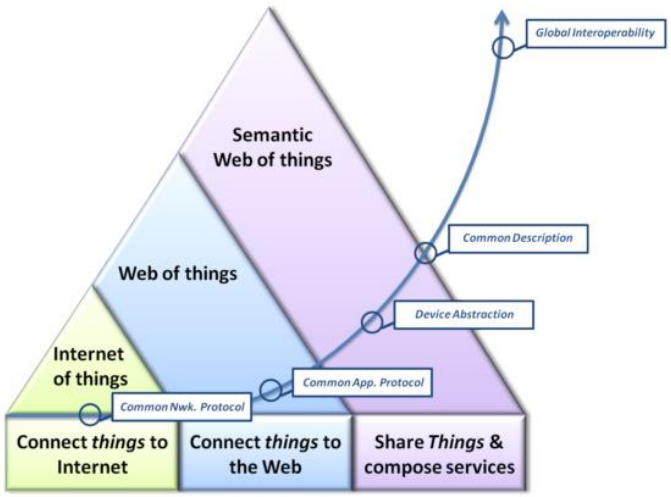



**Figure 2: Evolution of the market size from the Internet of Things to the Semantic Web of Things [8]**

Jara et al. [8] identified the three main interoperability challenges to be solved in order to complete the transition from the IoT to the SWoT:

- Heterogeneous device integration;
- Device abstraction;
- Syntactic and semantic interoperability.

In the same paper, the authors also address possible solutions for the challenges. For the support of heterogeneous devices, the IoT contains a common addressing space (such as IPv6, gateways or middleware), which allows the connection of all types of smart devices and things to the web in a homogeneous way.

By using web technologies, such as RESTful architecture, that define a communication layer where the resources are identified by a Universal Resource Identifier (URI), device abstraction can be attained. These URIs, besides providing abstraction, according to Jara et al. [8], are also able to "define semantic descriptions following structures such as Web Linking".

Regarding syntactic and semantic interoperability, two semantic web technologies have been playing an important role in the SWoT: Linked Data and Ontologies, which will be described in the next subsections.

### 2.2.1. Linked Data

The concept of Linked Data saw its inception in 2006, when Tim Berners-Lee published a set of four rules for "publishing data on the web, in a way that all published data becomes part of a single, global, data space" [10][11]:

1. Use URIs as names for things;
2. Use HTTP URIs so that people can look up those names;
3. When someone looks up a URI, provide useful information, using the standards (RDF, SPARQL);
4. Include links to other URIs so that they can discover more things.

These basic rules were named Linked Data Principles, and provide means for data publication on the Web and to connect it between different data sources, allowing the linkage of

data in different sources, with the employment of RDF and the Hypertext Transfer Protocol (HTTP) [12]. Linked Data is becoming an increasingly important part of the current smart city solutions, due to its interoperable nature and means of data representation. Consoli et al. [4], in the case of Catania, were able to develop a single Linked Open Data model, containing all the city's information, regarding its roads and urban transportation, public lighting maintenance waste management and urban fault reporting, by converting this heterogeneous information into RDF. The model was later complemented with a common ontology, developed in OWL, which described the city's business processes. The result was a semantic model that allowed the use of data extracted from different sources and unified under a shared semantic model, creating a platform for developing applications based on geo-location, real-time road traffic analysis and public transport management.

### 2.2.2. Ontologies

According to the World Wide Web Consortium, regarding the Semantic Web, vocabularies and their more complex formal counterpart, ontologies, "define concepts and relationships used to describe and represent an area of concern" [13]. The main roles of a vocabulary on the Semantic Web are, on one hand, to promote data integration, by resolving ambiguities on the terms of different data sets, and on the other hand, when new knowledge enables the discovery of new relationships [12].

Chandrasekaran et al [14] identify two major ontology classes regarding Knowledge Base Systems:

- **Ontology as vocabulary**, characterizing vocabularies as providing "a set of terms with which to describe the facts in some domain, while the body of knowledge using that vocabulary is a collection of facts about a domain";
- **Ontology as content theory**, characterizing ontologies as content theories, given their contributions to a domain – identifying the sorts of objects, their properties, as well as the relations between them.

In the same paper, the authors classify ontological analysis as a method for the clarification of the structure of knowledge. Within a domain, its ontology provides the central part of the information defined by it, and, without ontologies or the existence of the concepts defined by them, the existence of a vocabulary capable of representing the domain's knowledge is an unviable task.

According to Gyrard et al. [6], the rise of popularity of the ontologies is due to its capacity of easing the interoperability among applications, services, software and platforms and for describing domain knowledge. The authors identify the main benefits of using ontologies as (i) exchanging data between systems, (ii) providing interoperability among systems, (iii) designing and sharing knowledge, and (iv) simplifying operations.

The British Standards Institution developed a smart city concept model [15], able to describe all the entities, actors, occurrences and observations of a city through the use of ontologies. The goal of the model was to provide syntactical interoperability within a city's different sectors, solving the heterogeneity problems that usually result from the use of different terminologies to refer to the same actors within a system. The resulting model is able to provide complete syntactical interoperability among the sectors of a city, leaving doors open for the goal of semantic interoperability.

Gyrard et al. [16] proposed LOV4IoT, a dataset comprising circa 300 projects based on ontologies, referencing and classifying metrics such as (i) their applicative domains, (ii) the used sensors, (iii) the status and the information on the used ontologies, (iv) the reasoning that allows high level abstraction deductions, and (v) papers associated with the project [17]. Besides the classification of ontologies, the aim of the dataset was to promote the reuse of concepts in the development of new ones, with the final goal of attaining interoperability both among city sectors and among cities. The authors also proposed the Machine-to-Machine Measurement (M3) Framework, responsible for the generation of IoT application templates, based on the users' employed sensors and domains, and based on the ontologies present in LoV4IoT.

## 2.3. Data Model Evaluation

Over the years, the process of evaluating data models has been quite difficult to define and characterize. According to Maier [18], since data modeling has a varying nature, no actual general concept can be given for the evaluation of data models. According to Moody et al. [19], one of the biggest problems presented in the design of data modeling solutions is the substantial alternative designs that can be developed for a given problem. Hence, the process of data modeling should not be seen as a deterministic search process for the "correct data model", but as a "process of searching for alternative solutions. In order to get the best result, it is important to explore alternative models rather than simply adopting the first and most obvious solution." Maier [18] defended that the evaluation of data modeling needs to be focused on criteria reflecting

the quality of the modeling process and take into account different application scenarios of said modeling within an organization.

### 2.3.1. Definition of Criteria

William C. McGee [20], in his 1976 paper, devised a method for measuring a data model's quality, comprised of two main phases, firstly the identification of a set of model evaluation criteria (consisting in desirable attributes of a model), and secondly, the assignment of individual values to each item of the aforementioned criteria. Combining the weighted individual values of each criterion, one could easily calculate the model's suitability for a given data representation problem.

Accompanying this methodology, McGee defined twelve evaluation measurements, dividing them into two main groups: *use criteria*, measuring the model's ease of use, and *implementation criteria*, measuring the ease of the model's implementability as well as the efficiency of the resulting implementation.

Regarding *use criteria,* the following metrics were defined:

- **Simplicity**: A simple data model is described as having the "smallest possible number of structure types, composition rules and attributes";
- **Elegance**: A model is considered elegant if is capable of achieving a direct modeling capability with the least possible number of structures types, composition rules, and attributes;
- **Picturability**: All the structures defined in the data model should be able to be displayed in a graphic format;
- **Modeling directness**: The entities and relations presented in the model should have the maximum possible real-world counterparts;
- **Modeling uniqueness**: A model "should not provide equivalent direct modeling techniques", i.e., ideally, a model should provide only one way of modeling a certain situation that can be formulated with the terms defined in its knowledge base, in order to attain maximum performance;
- **Provision of schemas**: A model should contain structure schemas which allow data definition;

- **Implementation independence**: Data models usually contain convenient features for the implementers that are not necessary for the information modeling itself. Since these usually result in program dependence on them, they should be avoided;

- **Overlap with co-resident models**: A model should merge smoothly with other already-existing models;

- **Partitionability**: A model should contain structures that simplify the processes of partitioning data;

- **Non-conflicting terminology**: The terminology used by a data model should never conflict with already established terminology;

Regarding *implementation criteria*, two metrics were defined:

- **Proximity to the base model**: considered by the author as one of the most important model attribute from both an implementation and performance standpoint, a model should never be far from its implementation base, since whenever "structures of the model have exact counterparts in the base, it is unnecessary to develop procedures for transforming model structures into base structures and vice versa";

- **Applicability of safe implementation techniques**: A model should allow the use of already proved and well-understood implementation techniques.

Moody et al. [19], devised a comparison framework composed of five main constructs:

- **Qualities**: Defined as desirable properties or dimensions of value in data models, with the goal of the evaluation process being the maximization of the model regarding such properties;

- **Stakeholders**: In order to correctly classify a data model, one needs to have in mind the perspectives of owners and customers, as well as the data analyst's, the application users', and the data administrators';

- **Metrics**: The authors define "consistent and objective" ways of classifying the quality of a data model. While there is the possibility that several different measures exist for the same quality, an overall measurement may be attained, through the combination of the ratings for all the metric relating to it;

- **Weighting**: The attribution of weights to each of the model's qualities represent said quality's importance on the scope of the project;

- **Strategies**: The authors conceptualize strategies for the improvement of the data model. Instead of focusing solely on the model's qualities and flaws, one should also focus on defining methods for improving the models.

In the same paper, to complement the aforementioned framework, the authors provide a set of qualities to have in mind for the evaluation of the data models:

- **Simplicity**: Regarding the aspects of size and complexity of the data model, namely the number of contracts required, the authors note that, as a general rule, the simpler models are usually the best ones, since they provide more flexibility, easier implementations, and are easier to understand. According to the authors, "One of the major goals of data modeling should be to search for simpler and more powerful ways of representing the data, rather than just documenting user requirements". Proposed metrics for this quality are the measurement of the data model complexity, measured by the number of entities plus the number of relationships in the data model;

- **Completeness**: Regarding the ability of the data model to "meet all user information and functional requirements", complete and correct requirements information is considered to be a crucial prerequisite for a successful development of information systems. Proposed metrics for this quality consist of ratings given by user and industry specialists, process mapping and package comparison;

- **Flexibility**: To be considered flexible, a data model should be easily adaptable to changes in the requirements. This quality is rather important, since the "ability of a system to adapt to changes in its environment is widely considered as one of its most important characteristics". Metrics for the evaluation of this quality include senior management ratings, industry expert ratings, as well as data modeler ratings;

- **Integration**: To be defined as easily integrable, there needs to be a consistency of the information defined in the data model with the rest of the organization's data. Metrics for this quality consistent in a characterization of corporate data model conflicts;

- **Understandability**: this concept can be characterized as the "ease with which the concepts and structures in the data model can be understood by the users of the model". Evaluation metrics for this characteristic consist in ratings provided by users, data administrators, and application developers;

- **Implementability**: The implementability aspect of a data model can be defined as the ease with which the data model can be implemented, regarding aspects as time, budget, resource and technology constraints of the project.

### 2.3.2. Quantification of Qualitative Criteria

One of the most widely used methods of quantification of qualitative measurements is the employment of a Likert Scale. This psychometric tool was first devised in 1932 by psychologist Rensis Likert as a means to "measure 'attitude' in a scientifically and validated manner" [21], where attitude can be defined as a preferential way of "behaving/reacting in a specific circumstance rooted in a relatively enduring organization of belief and ideas" [21].

The Likert scale consists of a series of four or more questions (Likert items), each having a set of possible responses (originally five options), usually ranging from strong approval to strong disapproval of the concept. These responses are then combined in order to create an attitudinal measurement scale [22], objectively reflecting the test subject's opinion on a certain phenomenon.

Joshi et al [21] identified three principles for the correct construction of a Likert scale

1. The minimum score one can secure for the first three items of a scale is 3
2. Each numeral (response option) conveys the same meaning in all of the survey's items
3. All the scale items can be clubbed while satisfying the content and criterion validity

In its majority, due to the restrictions implied by the third principle, a new concept was created, in order to apply the same principles of the Likert Scale to a set of non-related questions. This kind of scale is called a Likert-type scale, and is used in situations where there is no attempt by the researcher to combine the responses from the items into a composite scale, but rather when the primary objective of the researcher is to study the opinions, beliefs or feelings a certain phenomenon causes in the test audience, instead of the generalization of the stance of the participants.

In their 2015 paper, Joshi et al. [21] also differentiate the process of data analysis based on the type of scale used. When dealing with Likert-type scale data, the values assigned to the items express a *greater than* relationship, with the size of the comparison being undefined. Given the uncertainty of this measurements, the authors consider this type of data to fall under the ordinal measurement scale. Due to this fact, the authors recommend the use of descriptive statistic tools, such as "modes or median values for central tendency and frequencies for variability".

On the other hand, when dealing with Likert scale data, the items are created by calculating a composite score from four or more items through the use of a sum or a mean, the authors state that the composite score should be analyzed at the interval measurement scale. For the interpretation and analysis of the results, the authors recommend the use of descriptive statistical tools, such as "the mean for central tendency and standard deviations for variability".

## 2.4. Conclusions

Great efforts are currently being made in the fields of data interoperability within IoT, and more precisely smart city systems.

The concept model developed by the British Standards Institute, mentioned in Section 2.2.2, provides a good starting point for the task in hands, since the achievement of syntactic interoperability is the basis for the achievement of semantic interoperability. However, and according to the authors, "Sharing data across a city requires more than the interoperability covered by the SCCM." [15]. The LOV4IoT solution, presented by Gyrard et al. [16], also offers some advancements in the fields of data interoperability, by being able to aggregate ontology based projects among smart city sectors and finding common grounds between them. However, the platform could not be found, thus providing no contributions for this paper.

The solution for the problem previously mentioned will take into account the research and study of currently available data models for the IoT, and the construction of a comparison framework in order to categorize them. While the works developed by Moody [19], Maier [18] and, most importantly, by McGee [20] provide a good stepping stone for the evaluation of data models, they can be considered dated and not focused on the intricacies of the problems presented by IoT systems and subsequent communication protocols. In order to cover these unaddressed dimensions, additional criteria will be devised for the development of the framework.

# 3. Data Models for the IoT

This section presents a preliminary suitability analysis for Citibrain's platform of the seven discovered data models - FIWARE [23], OGC SensorThings [24], CitySDK[25]–[27], OData[28], W3C Generic Sensor [29], IPSO Smart Objects [30] and oneIoTa [31]. The analysis starts by identifying the abstraction level of the models. This property can have one of two possible values: (i) Instantiated, if a set of predefined data models for certain metrics and observations are provided, and (ii) Abstract, if only one model is provided, which only defines the entities of the system and their relationships. In a second step, if the models are instantiated, three dimensions are studied:

- The smart city sectors they apply to;
- The list of the available predefined models;
- The expandability of the models, i.e., if it is possible to develop non-provided models.

The main goal of the analysis is to determine if the data models which models are instantiated, and of this group, which ones dispose of structures capable of representing information related to the fields of air quality management, waste management, parking management and traffic management, which represent the four vertical solutions developed by Citibrain.

Table 2 presents the results of the analysis.

**Table 2: Comparison of the data models' abstraction level**

| Name | API | Abstraction | Expandable | Mobility | Waste | Environment |
|---|---|---|---|---|---|---|
| **FIWARE** | NGSI v2 | Instantiated | Yes | No | Yes | Yes |
| **SensorThings** | SensorThings | Abstract | N/A | N/A | N/A | N/A |
| **CitySDK** | CitySDK | Instantiated | Yes | Yes | No | No |
| **OData** | Unspecified | Abstract | N/A | N/A | N/A | N/A |
| **GenericSensor** | GenericSensor | Abstract | N/A | N/A | N/A | N/A |
| **Smart Objects** | Unspecified | Abstract | N/A | N/A | N/A | N/A |

- Fiware provides instantiated models for all Citibrain's sectors, is expandable and provides guidelines for the development of new data models. These data models are considered suitable for Citibrain.

- CitySDK provides instantiated models, fitting only one of Citibrain's sectors – mobility, and, even within that sector, provides data models for none of Citibrain's vertical solutions. Although expandable, there is a need for the registration of the city's municipality on the platform's website, which results in a possibly prolonged bureaucratic process. Due to the closed nature of the platform, it can be considered a bad solution for Citibrain.

- oneIoTa provides instantiated data models, fitting, however, only one of Citibrain's sectors – Air Quality Management. Since the models are expandable, the platform can be considered a good solution for Citibrain, and data model contributions can be considered a good complement for the research and development aspects of this project.

- SensorThings, Generic Sensor, OData and Smart Objects are abstract models, meaning all four can be used in Citibrain's context.

- Since during the research process developed in section 3 little documentation was discovered for the OData data models, with the majority of it being oriented towards business management, it can be considered a bad fit for Citibrain.

Bearing in mind the aforementioned conclusions, five models have been selected for further study: FIWARE, SensorThings, Generic Sensor, Smart Objects, and oneIoTa.



# 4. Comparing Data Models

Upon the completion of the preliminary suitability analysis, a deeper analysis of the selected models, regarding their performance is needed. For this analysis, the process is divided into two main analysis categories: quantitative and qualitative, whose selection of metrics and calculation will be explained, with detail, in the next subsection.

## 4.1. Formulation of Evaluation Criteria

The evaluation and comparison of data models is divided into two main components: (i) qualitative, referring to more non-measurable and subjective qualities of the data models, based on the research carried in section 2.3; and quantitative, corresponding to the more easily

measurable qualities of the data models, related, in their majority to aspects of latency and communication loads.

### 4.1.1. Qualitative Evaluation

For the qualitative analysis, a set of nine criteria was chosen. Of these criteria, seven metrics were selected from the analyzed publications, since they were deemed the most relevant of all the studied metrics, and representative of the desirable qualities of a data model: (i) overlap with co-resident models, (ii) simplicity, (iii) completeness, (iv) implementability, (v) flexibility, (vi) understandability, and (vii) easiness of development.

In order to better complement the analysis, two additional metrics were added to the qualitative analysis: (i) amount and quality of available documentation, and (ii) the existence of implementation tools, in order to achieve a more complete evaluation.

The definition and quantification process of the selected metrics are as follows:

- **Integration/Overlap with co-resident models**: calculated as the consistency between its information structure and Citibrain's database's. In order to provide a more illustrative value, the metric is to be calculated as the amount of Citibrain's base model entities the new data model is able to represent;

- **Simplicity**: by the sum of the number of described entities plus the number of relations present in the model;

- **Completeness**: calculated as a ratio between the simplicities of the data model and Citibrain's base model;

- **Implementability:** calculated as the number of hours spent in implementing the data model;

- **Flexibility**: calculated as a subjective analysis of the model's ability to be compliant with changes in the original database information structure, quantified through the use of a Likert-type survey;

- **Understandability**: calculated as a subjective analysis of the ease with which a user with reasonable knowledge in data modeling and willingness to study the information will have to understand the concepts, structures and relations present in the model, quantified through the use of a Likert-type survey;

- **Easiness of development**: calculated as a subjective analysis of the implementation experience, in the form of a grading system, quantified through the use of a Likert-type survey;

- **Amount and quality of available documentation**: calculated as a quantitative study on each data model, based on the following seven metrics: (i) Does the platform provide a specification for its data model? (ii) Do implementation guidelines exist? (iii) Are there any tutorials provided? (iv) Does a repository exist? (v) Are examples given of the models' implementation? (vi) Does a community exist? If so, is it responsive? (vii) Are there any questions related to the models on *Stack Overflow?* If this is the case, do they have accepted answers?

- **Implementation Tools:** This metric will be calculated by the model's platform ability to answer the following questions: (i) Does the platform provide any kind of SDKs? (ii) Are there any libraries available? (iii) Do any validation tools exist?

For the quantification of the last two metrics, the model is to be awarded one point for each question it can answer.

### 4.1.2. Quantitative Evaluation

Regarding the quantitative analysis, four criteria representative of desirable qualities in a data model for the IoT were selected:

- **Temporal Overhead**: determined as the amount of time spent on the conversion from the original database information structure to the one present in the data model;

- **Memory Overhead**: determined as the increase in the computer's memory resources noted during the information conversion;

- **Representational Overhead**: calculated as the size, in bytes, of both the representation of a dataset and the representation of an individual registry for each data type to consider;

- **Request Overhead:** measured as the time, in milliseconds, between the sending of the API request and the receiving of the system's answer.

## 4.2. Quantification of Evaluation Criteria

In order to quantify the metrics defined in the previous subsection, a formula was devised in order to translate the performed evaluation into a single value, representing the model's

suitability for Citibrain's IoT platform. The general formula of the data model's quality (MQ) is represented by:

$$MQ = W_{Qt} * QtEv + W_{Ql} * QlEv$$

Where $W_{Qt}$ and $W_{Ql}$ represent the weights, and QtEv and QlEv represent the measures of the quantitative and qualitative evaluations. In this paper, and without loss of generality, it was chosen to give the quantitative evaluation a higher weight, since it is more objectively measurable. Thus, throughout the rest of this work $W_{Qt}$ is given a value of 0.7 and $W_{Ql}$ a value of 0.3.

### 4.2.1. Quantitative Metrics

The Quantitative Evaluation is calculated with the values of the memory efficiency (M.E.), temporal efficiency (T.E.), representational efficiency (R.E.) and request efficiency (Rq.E.), introduced in the conversion of the data from the base model to the new data model.

$$QtEv = W_{ME}* M.E. + W_{TE} * T.E. + W_{RE} * R.E. + W_{RqE}* Rq.E.$$

As all the efficiency values can be measured with the same precision, and all affect the total overhead in the same manner, for the performed evaluation, the same weight should be attributed to all of them - 0.25.

Since the final values of the formula vary between 0 and 1, with 1 representing the most suitable model possible, the normalized values of the overheads are subtracted from 1, thus representing the data models' efficiency.

For the measurement of the memory, temporal and request overheads, ten measurements are made for the conversion of the entirety of the database registries of each vertical solution. An average of the ten values is then calculated, and the results of all vertical solutions' mean times are added. The request overhead is calculated by summing the sizes of all JSON objects provided by the API for a request of the bulk of the data contained in the database. In order to normalize the values, the total overhead of each metric for each model is divided by the sum of all model's corresponding overheads and then subtracted from 1. The calculation of the request overhead can be consulted in the following formula:

$$Rq.O.^{Model\ X} = 1 - \Sigma\ (SizeOf\ (JSONObjects)) / \Sigma\ (Rq.O.^{\ Studied\ Models})$$

### 4.2.2. Qualitative Metrics

The qualitative evaluation represents the quantification of the more difficultly measured characteristics that define the quality of a data model. Within this group of criteria, two sub-groups can be identified: one containing the more easily measurable metrics, designated as

Objective Qualitative Criteria (Obj), and one containing the more abstract metrics, designated as the Subjective Qualitative Criteria (Subj):

$$QlEv = W_{Obj} * Obj + W_{Subj} * Subj$$

Where $W_{Obj}$ and $W_{Subj}$ represent the weights attributed to each set of metrics. For this paper, following the same principle and the weight distribution between the qualitative and the quantitative analysis, the objective analysis was attributed a bigger weight, of 0.75, and the subjective analysis the smaller (0.25).

The objective qualitative criteria is defined by six metrics: (i) the amount of available documentation and available support (Doc), (ii) the model's overlap with Citibrain's current information structure (Ovrlp), (iii) the simplicity of the model's implementation (Simp), (iv) the model's completeness (Comp), (v) the model's implementability (Imp), and (vi) the existence of implementation tools (ITools):

$$Obj = W_{Doc} * Doc + W_{Ovrlp} * Ovrlp + W_{Simp} * Simp + W_{Comp} * Comp + W_{Impl} * Impl + W_{IT} * ITools$$

Where $W_{metric}$ represents the weight of each metric. For the evaluation present in this paper, all the metrics were attributed the same weight - 1/6.

The **documentation** value is measured through a point system, where a point is attributed to the model if it satisfies each of the questions asked in the previous section regarding documentation and available support:

1. Does the platform provide a specification for its data model?
2. Do implementation guidelines exist?
3. Are there any tutorials provided?
4. Does a repository exist?
5. Are examples given of the models' implementation?
6. Does a community exist? If so, is it responsive?
7. Are there any questions related to the models on *Stack Overflow?* Do they have accepted answers?

After calculating the value for all the studied models, the result is normalized, by diving it by the maximum possible value - 7.

The **simplicity** value is calculated by adding the number of entities in the models and the relations and dividing the result by the maximum value observed. The value is then subtracted from 1.

The **completeness** value is calculated by dividing the sum of entities and relations of the model by the sum of entities and relations of the original base model and subtracting the result value from 1.

The **implementability** value is calculated by dividing the number of hours spent implementing each of the models, by the total number of hours spent in the implementation of all the models. Since the model with the least implementation hours is more desirable, the result is also subtracted from 1.

The **implementation tools** value is calculated in similar fashion to the documentation metric. The model gets one point for each question asked in the previous section it can satisfy, and is then normalized.

The **overlap with current resident models** is calculated by mapping the base model's entities to the data models'. The result is then normalized.

Regarding the subjective criteria, three metrics were chosen: the easiness of implementation (EoI), the understandability (Undst), and the flexibility (Flex) of the model, with the final value calculated as is shown in the following formula:

$$Subj = W_{EoI} * EoI + W_{Undst} * Undst + W_{Flex} * Flex$$

For the developed analysis in this paper, all the subjective criteria were given the same weight of 1/3. Due to the highly subjective nature of these three metrics, the values were obtained through the elaboration of a Likert type survey, consisting of three questions, with five possible answers, ranging from Strongly Disagree to Strongly Agree:

1. The Data Model is easy to implement
2. The Data Model's concepts and entities are easy to understand
3. The Data Model can be considered flexible to changes in the

The possible answers for these statements were (i) Strongly Agree (with a value of five points), (ii) Agree (with a value of four points), (iii) Neither Agree or Disagree (with a value of three points), (iv) Disagree (with a value of two points), and (v) Strongly Disagree (with a value of one point). The answer's values are then added and divided by the product of the number of questions and the maximum possible value of the answer (15, in this case) [21].

The full extent of the formula can then be calculated as follows:

$$M.Q = W_{Qt} * (W_{ME} * M.E + W_{TE} * T.E. + W_{RE} * R.E. + W_{Rq.E.} * Rq.E.) +$$

$W_{Ql}$ * ($W_{Obj}$ * ($W_{Ovrlp}$ * Ovrlp + $W_{Doc}$ * Doc + $W_{Simp}$ * Simp + $W_{Comp}$ * Comp + $W_{Imp}$ * Imp + $W_{ITools}$ * ITools) + $W_{Subj}$ * ($W_{EoI}$ * EoI + $W_{Undst}$ * Undst + $W_{Flex}$ * Flex))

Applying the chosen weights for the performed evaluation in this thesis, the final value of the model's quality can be calculated as follows:

M.Q = 0.7 (1/4 * M.E + 1/4 * T.E. + 1/4 * R.E. + 1/4 * Rq.E.) + 20
0.3 * (0.75 * (1/6 * Ovrlp + 1/6 * Doc + 1/6 * Simp + 1/6 * Comp + 1/6 * Imp + 1/6 * ITools) + 0.25 * (1/3*EoI + 1/3*Undst + 1/3*Flex))

## 4.3. Final Considerations on the Evaluation

Regarding the quantitative analysis, all the measurements were made in a dedicated DS2 Microsoft Azure Ubuntu virtual machine[11], with the following specifications:

- 2 CPU cores
- 7 GiBs of RAM;
- 100 GBs of disk space;
- local SSD drive with 14 GiB;

The values of the temporal overheads, and subsequently efficiency, were measured by the Python function *TimeIt*[12], responsible for measuring the execution time of code snippets. The values of the memory overheads (and subsequently efficiency) were measured through the use of Python *resource.usage(ru_maxrss)*[13] function, responsible for measuring the resource usage information, in this case, the maximum resident set size.

In spite of the devised framework being designed to be a generic approach to the ranking and evaluation of data models for the IoT, it was designed with the notion of the limitations provided by the size of the test group, which consisted of only one subject. With these notions in mind, some considerations can be made regarding the attributed weights and the choice of the metrics:

[11] docs.microsoft.com/pt-pt/azure/virtual-machines/windows/sizes-general

[12] docs.python.org/2/library/timeit.html

[13] docs.python.org/2/library/resource.html

- The metrics involving the number of hours of implementation, and the subjective measurements should be evaluated by more than one developer, in order to make the results more general. However, there was a lack of means to set up a study in such conditions, in this work.

- If the test group is comprised of several subjects with experience in data modelling and data model evaluation, different weights might be attributed to both the quantitative and qualitative analysis, reaching up to the same weight.

- Some other Likert Items can be included in the subjective portion of the formula, in order to attain a more complete idea of the developers' experience with the data model. If this is the case, in order to avoid *respondent fatigue*, an even scale should be used. This was not the case with the conducted study, due to the small amount of items present in the developed survey.

Although the objective metrics of the formula should always have a bigger impact in the final value, given the easily measurable nature, it is always important to have in mind that, as stated by Maier [18], the quality of the data modelling experience provides an equally important notion as quality of the data model itself. This last notion is described by the author as a "method, which supports the process of adaptation, standardization and integration in the development of application systems".

## 5. Results

Table 4 displays the full extent of the obtained results, upon the evaluation of the selected data models.

**Table 2: Data Model Suitability Evaluation**

| | Fiware | W3C Generic Sensor | IPSO Smart Objects | oneIoTa | OGC SensorThings |
|---|---|---|---|---|---|
| Temporal Efficiency | 0.84 | 0.81 | 0.66 | 0.73 | 0.96 |
| Memory Efficiency | 0.86 | 0.82 | 0.80 | 0.72 | 0.79 |
| Representational Efficiency | 0.92 | 0.92 | 0.94 | 0.28 | 0.94 |
| Request Efficiency | 0.85 | 0.79 | 0.93 | 0.58 | 0.84 |
| **Quantitative Analysis** | 0.87 | 0.84 | 0.83 | 0.58 | 0.88 |
| Entity Similarity | 1.00 | 0.56 | 1.00 | 1.00 | 0.78 |
| Documentation | 0.95 | 0.43 | 0.58 | 0.29 | 0.57 |
| Simplicity | 0.44 | 0.44 | 0.69 | 0.19 | 0.00 |
| Completeness | 0.44 | 0.44 | 0.69 | 0.19 | 0.00 |
| Implementability | 0.80 | 0.92 | 0.92 | 0.74 | 0.62 |
| Implementation Tools | 1.00 | 0.33 | 1.00 | 0.33 | 0.67 |
| Likert Survey | 0.93 | 1.00 | 0.80 | 0.87 | 0.80 |
| **Qualitative Analysis** | 0.81 | 0.64 | 0.81 | 0.56 | 0.53 |
| **Data Model Quality** | 0.85 | 0.78 | 0.82 | 0.57 | 0.78 |

Observing the results, some conclusions can be drawn regarding the impact of the level of abstraction on the implementation of the data models:

- Instantiated data models need bigger instantiation periods, but often result in more complete and structured information. Such case was Fiware's. Despite the similar structure to the base model, the data model's implementation time was the third largest. However, the data model's information structure allowed all the relevant information regarding Citibrain's verticals to be stored in few tables with a relatively low number of external keys, meaning low granularity in the stored information;

- Abstract models present the advantage of being able to store all current information in the same database tables, as well as additional new information. This translates to the needlessness of applying any changes to or creating new entities in the data model towards the development of new vertical solutions by the smart cities platform.

Regarding the performed analysis, some considerations can also be made.

Overall, Fiware displayed better performance, and ended up being selected for the server implementation of the developed dashboard. Due to the current information structure of Citibrain's data model, the result can be considered positive, and the transition to the new data model a rather simple and straightforward process.

In a close second came IPSO Smart Objects. Despite the simplistic nature of its information structure, the data model revealed itself to be a powerful tool, presenting low implementation times, low overheads, as well as high simplicity and completeness values. Unlike Generic Sensor, IPSO Smart Object's simplistic data model didn't reveal itself as a flaw, with the model being able to represent all Citibrain's entities and relations. The model also achieved the best values in the implementability metric, and scored very similar overhead measurements to Generic Sensor. With the advantages and no disadvantages of the simplicity of its information representation, Smart Objects can be considered a good candidate for Citibrain.

In third and fourth place, displaying very similar end results came Generic Sensor and SensorThings. While Generic Sensor presented an overly simplistic architecture, not being able to represent all the base model's entities and their relations, the opposite problem can be noted in SensorThings' structure. Due to the high granularity and redundancy of the information structure, the data model ended up forcing the inclusion of duplicated information in the database, in many cases having the *Sensor* and *Thing* entities representing the same real life entity. Even despite SensorThings' score of the lowest level in the simplicity metric, the model was not able to represent all the base model's entities and their relations, scoring the second lowest value in the

*Overlap with co-resident models* metric. However, in spite of its high level of granularity of its information structure and existence of repeated information, SensorThings presented the best results of the qualitative analysis. Generic Sensor also scored one of the lowest support and documentation values, while having no implementation guidelines or tutorials available. On the other hand, due to the data model's simplicity, Generic Sensor scored one of the best results in the overheads measurements. Overall, none of the two models can be considered a good solution for Citibrain's platform.

Finally, oneIoTa scored the lowest overall results of all the analyzed models. Despite being implemented with a similar architecture to Fiware, the models presented a higher level of granularity, making distinctions between things and locations, which ended up creating some overheads regarding the communication and representation aspects. Another big drawback from the data model is the overheads created in the serialization and implementation processes, due to the need of the inclusion of verbose headers and footers in each model. Due to being a data modelling tool, and the platform's incentive for its users to create and submit new data models, oneIoTa was able to represent all of Citibrain's entities and relationships. Achieving a high score on the subjective analysis, the data model tool proved itself understandable and easy to implement. The data model tool was also the least documented of the studied data models, having no community presence, and only disposing of a repository for the implemented data models, which also serves as examples.



# 6. Conclusions

The main goal of this paper was to perform an analysis on currently available data models for the IoT, with the intent of finding a model that translated into a good solution for Citibrain's platform. In order to achieve such goal, a set of criteria had to be designed in order to characterize what makes a good data model, and, most importantly, what makes a data model the right fit for an IoT system.

In order to answer these questions, a research on data model quality was performed, and a set of criteria was selected. Such criteria were then used to create a data model evaluation framework capable of translating a data model's suitability for a platform into a single value for easy comparison with other models. This framework was then used to test a set of data models' suitability for Citibrain's platform and a solution was found.

All the proposed objectives were met, and the presented solution granted a meaningful contribution to the field of the Internet of Things, since no other data model comparison frameworks directed to the IoT were found during the research stage. Although the analysis was directed to Citibrain's case, it can be considered generic enough to be applied to other specific

cases. The development of new criteria arose from the lack of data model evaluation metrics directed to the Internet of Things. Where some evaluation dimensions and criteria could be found in the research activities, a lack of metrics directed exclusively for the IoT could not be found. Due to the constrained nature of the Internet of Things, some specificities had to been taken into account. Metrics such as the response times of the server containing each data model, or the load introduced in the communication network caused by the representation of each model have a great impact in machine communication, bearing a great importance within the framework's evaluation.

## 6.1. Analysis Limitations

The biggest limitation of the presented study is the size of the test group for the developed analysis framework, consisting of only one subject. Based on that fact, some considerations can be made regarding the limitations of the analysis.

- The measured overheads were extremely limited to the implementation method. In an optimal setting, the analysis would be made by averaging the results of the implementations of several developers, in order to achieve a more conformist result;

- The same principle applies, with even more weight, to the subjective part of the qualitative analysis. Given the test audience being composed of just one subject, the obtained results cannot translate into a realistic evaluation of the measured metrics.

For the analysis to have an actual impact, the evaluation should be performed by multiple experts, in order to avoid biases by individual reviewers. As pointed by Moody [19] , other possible way would be to have different experts selected to "apply the metric independently and the results combined into a single rating".

Another limitation falls within the calculation of the request overhead, which only measures the server's response time of the bulk of the registries of a single entity. In the case of abstract models, the information of a single sensor measurement can be stored across several database tables, and for a correct measurement of the request overhead, the additional queries performed by the server in order to aggregate all the information, should also be taken into account.

## 6.2. Future Work

Some future work includes the study and survey of the data models' implementation by a wider test audience. Due to the impact of the conversion software in both the qualitative and the

quantitative analysis, metrics such as temporal overhead, memory overhead and implementability, the average of the results of a wider group's implementation is imperative for more precise outcomes.

Also when dealing with a bigger audience, the formula's weights can be modified, in order to give a bigger importance to the subjective metrics, as well as the inclusion of more Likert items in the survey. In order to achieve this, a web platform can be created, enabling IoT developers to share their experiences with the implementation of the data models.

Regarding the performed analysis, some improvements can also be made, such as the inclusion of additional data models in the study, and the inclusion of other metrics specific for Citibrain's platform, such as supported communication protocols. Another improvement for more precise results, would be the development of a complete API for all the models, for a more correct measurement of the request overhead.

# References


[1] M. Batty *et al.*, "Smart cities of the future," *Eur. Phys. J. Spec. Top.*, vol. 214, no. 1, pp. 481–518, 2012.

[2] S. Consoli *et al.*, "A Smart City Data Model based on Semantics Best Practice and Principles," *Proc. 24th Int. Conf. World Wide Web Companion*, pp. 1395–1400, 2015.

[3] S. Bischof, A. Karapantelakis, A. Sheth, A. Mileo, and P. Barnaghi, "Semantic Modelling of Smart City Data," *W3C Work. Web Things Enablers Serv. an open Web Devices*, pp. 1–5, 2014.

[4] S. Consoli, M. Mongiovì, D. Reforgiato, S. Peroni, and A. Gangemi, "The Role of Semantics in Smart Cities Producing Linked Data for Smart Cities : the case of Catania," vol. 1, pp. 1–5, 2014.

[5] A. K. B, Z. Jan, A. Zappa, and M. Serrano, "Interoperability and Open-Source Solutions for the Internet of Things," vol. 10218, pp. 20–35, 2017.

[6] A. Gyrard, M. Serrano, and G. A. Atemezing, "Semantic web methodologies, best practices and ontology engineering applied to Internet of Things," *2015 IEEE 2nd World Forum Internet Things*, pp. 412–417, 2015.

[7] E. R. C. on the I. of Things, "Internet of Things - IoT Semantic Interoperability: research challeges, best practices, recommendations and next steps," vol. 1, p. 48, 2015.

[8] A. J. Jara, A. C. Olivieri, Y. Bocchi, M. Jung, W. Kastner, and A. F. Skarmeta, "Semantic Web of Things: an analysis of the application semantics for the IoT moving towards the IoT convergence," *Int. J. Web Grid Serv.*, vol. 10, no. 2/3, pp. 244–272, 2014.

[9] T. I. M. Berners-lee, J. Hendler, and O. R. A. Lassila, "The Semantic Web will enable machines to," *Sci. Am.*, no. May, pp. 1–4, 2001.

[10] C. Bizer, T. Heath, and T. Berners-Lee, "Linked data-the story so far," *Int. J. Semant. Web Inf. Syst.*, vol. 5, no. 3, pp. 1–22, 2009.

[11] T. Berners-lee, "Linked Data," 2006. [Online]. Available: https://www.w3.org/DesignIssues/LinkedData.html.

[12] C. Bizer *et al.*, "Linked data on the web," *WWW2008 Work. Linked Data Web*, pp. 1265–1266, 2008.

[13] World Wide Web Consortium, "Ontologies." [Online]. Available:

https://www.w3.org/standards/semanticweb/ontology. [Accessed: 25-Jan-2017].

[14] B. Chandrasekaran, J. R. Josephson, and V. R. Benjamins, "What aro ontologies, and why do we need them?," *IEEE Intell. Syst. Their Appl.*, vol. 14, no. 1, pp. 20–26, 1999.

[15] B. S. Institution, "BSI Standards Publication Smart city concept model – Guide to establishing a model for data interoperability," pp. 1–56, 2014.

[16] A. Gyrard, C. Bonnet, K. Boudaoud, and M. Serrano, "LOV4IoT: A second life for ontology-based domain knowledge to build Semantic Web of Things applications," *IEEE 4th Int. Conf. Futur. Internet Things Cloud*, 2016.

[17] A. Gyrard and M. Serrano, "Connected Smart Cities : Interoperability with SEG 3 . 0 for the Internet of Things," no. 2, pp. 1–7, 2016.

[18] R. Maier, "Evaluation of Data Modeling," 1995.

[19] D. L. Moody and G. G. G. Shanks, "What Makes a Good Data Model? Evaluating the Quality of Entity Relationship Models," *Proc. the13th Int. Conf. Entity-relationsh. Approach*, pp. 94–110, 1994.

[20] W. C. McGee, "On user criteria for data model evaluation," *ACM Trans. Database Syst.*, vol. 1, no. 4, pp. 370–387, 1976.

[21] A. Joshi, S. Kale, S. Chandel, and D. Pal, "Likert Scale: Explored and Explained," *Br. J. Appl. Sci. Technol.*, vol. 7, no. 4, pp. 396–403, 2015.

[22] H. N. J. Boone and D. A. Boone, "Analyzing Likert data," *J. Ext.*, vol. 50, no. 2, p. 30, 2012.

[23] FIWARE, "About us," 2016. [Online]. Available: https://www.fiware.org/about-us/. [Accessed: 06-Jan-2017].

[24] OGC, S. Liang, A. C. Huang, T. Khalafbeigi, and K. Kim, "OGC SensorThings API Candidate Standard Part 1: Sensing," pp. 1–88, 2015.

[25] CitySDK, "Issue reporting / Open311 API." [Online]. Available: https://www.citysdk.eu/citysdk-toolkit/using-the-apis/open311-api/. [Accessed: 12-Jan-2017].

[26] CitySDK, "Tourism API." [Online]. Available: https://www.citysdk.eu/citysdk-toolkit/using-the-apis/tourism-api/. [Accessed: 12-Jan-2017].

[27] CitySDK, "Linked Data API." [Online]. Available: https://www.citysdk.eu/citysdk-toolkit/using-the-apis/linked-data-platform. [Accessed: 12-Jan-2017].

[28] M. Pizzo, R. Handl, and M. Zurmuehl, “OData Version 4.0 Part 1: Protocol,” *OASIS Stand.*, no. February, pp. 1–69, 2014.

[29] World Wide Web Consortium, “Generic Sensor API,” 2016. [Online]. Available: https://www.w3.org/TR/2016/WD-generic-sensor-20160830/. [Accessed: 28-Jan-2017].

[30] J. Jimenez, M. Koster, and H. Tschofenig, “IPSO Smart Objects,” 2016.

[31] O. C. Foundation, “oneIoTa User Guide,” 2016.