4

Development Tools for IoT Analytics Applications

John Soldatos and Katerina Roukounaki

Athens Information Technology, Greece

4.1 Introduction

The proliferation of IoT analytics applications has recently created a need for tools and techniques that could support developers in the task of producing and deploying IoT analytics services. In principle, IoT analytics development tasks can be supported by readily available tools for IoT applications and their combination with tools and techniques for data mining and analytics. In particular, IoT development tools undertake to collect and appropriately pre-process data streams stemming from IoT systems (including "live" high-velocity data streams), while conventional data analytics tools can be used to analyze the information contained in these data streams towards extracting knowledge. Hence this combination brings together the IoT and BigData worlds, thus facilitating developers in the task of implementing and deploying IoT analytics applications.

This chapter is destined to present this blending of IoT development tools and data analytics tools. In particular, the chapter is devoted to the presentation of sample tools for the development of IoT analytics applications and more specifically the development tools of an IoT platform which has been recently developed as part of the FP7 VITAL project. These tools support IoT development functionalities such as discovery of data streams from IoT systems, filtering of data streams in order to economize on bandwidth and storage resources, as well as semantic unification of heterogeneous streams in order to facilitate the unified processing of diverse data sources. The importance of such functionalities for IoT analytics applications is adequately described in the scope of other chapters of this book, along with specific technology solutions

for their implementation. The present chapter considers these functionalities as part of the presented development tools infrastructure, which leverages middleware services (e.g., data streams discovery and filtering) of the VITAL platform. Therefore, the chapter introduces these middleware services as well, along with their positioning in the overall architecture of the VITAL platform.

The VITAL development tools are based on the popular Node-RED tool for IoT applications, which has been customized to the needs of the VITAL platform. The customization of the Node-RED tool included also the enhancement of data mining and data analytics functionalities, which are illustrated in the scope of this chapter. Along with the VITAL development tools, the VITAL platform provides also a tool for managing IoT resources (including IoT data sources and data streams), including configuration, security and SLA (Service Level Agreement) management functionalities. The latter can greatly facilitate the monitoring of IoT analytics applications and can be used in conjunction with the VITAL development tools. Therefore, we also present the VITAL development tools as an integral element of the wider suite of tools that support developers in the production of IoT analytics applications. The development and management tools are bundled in an integrated development environment, which is accessible over the web and from a single entry point.

Overall, the chapter is structured as follows: The next paragraph discusses relevant work on development tools for IoT analytics. Following chapters illustrate the VITAL architecture and the middleware services that are used in order to support the functionalities of the tools in the scope of the VITAL platform. Along with these functionalities, the chapter discusses the VITAL development tools with particular emphasis on their add-on features which enhance Node-RED. The discussion includes also insights on the limitations of the development tools, which could be remedied as part of future work. Moreover a dedicated section is devoted to the description of the VITAL management environment. Indicative applications are finally presented in order to illustrate the added-value of the tools and the productivity boost that they can offer to large number of developers of IoT analytics applications.

4.2 Related Work

The provision of development environments for IoT analytics has its roots on tools and techniques for the development for IoT applications and data analytics. IoT development tools provide the means for interfacing to IoT systems towards collecting, filtering and fusing IoT data streams.

At the same time, data analytics environments provide the means for developing and executing data analytics algorithms.

Early IoT development tools have been introduced as part of WSN (Wireless Sensor Network) platforms (e.g. [1, 2]) and RFID (Radio-frequency Identification) platforms (e.g. [3]). Recently we have witnessed the emergence of integrated development environments and tools for wider classes of IoT applications, including visual modelling tools following Model Driven Architectures (MDA) (e.g. [4, 5]).

There have also been IoT development environments associated with mainstream IDE projects, such as Eclipse Kura, which is an Eclipse IoT project that provides a framework for M2M service gateways (i.e., devices that act as mediators in the machine-to-machine and the machine-to-Cloud communication). Kura facilitates the development, deployment and remote management of M2M applications and its use requires only the installation of an Eclipse plugin on the developer's machine. It is based on Java and OSGi, the dynamic module system for Java, and it can be used to turn a Raspberry Pi or a BeagleBone Black into an IoT gateway.

Node-RED is another open-source project that is focused on IoT. This project is reused and extended as part of the prototype implementation presented in this chapter. It is described in the following paragraph in order to facilitate the understanding of the approach and the related implementation.

Integrated Cloud Environments (ICEs) have come to change this workflow, by turning development environments from products into services. ICEs are essentially IDEs that are usually web accessible, and that leverage the Cloud into the software development lifecycle. In order to use an ICE, developers do not need to install any more tools on their machines; all they need to do is log into a web site (that acts as the entry point to the ICE), and start using it. In this case, most of the tasks take place in the Cloud; some ICEs use the Cloud even to store the developers' code.

While IoT tools provide the means for interfacing to data sources towards accessing, processing and combining data streams, they do not typically offer capabilities for analyzing IoT data. Therefore, their use for IoT analytics requires their integration with data analytics libraries and tools such as:

- The Technical Analysis library (http://ta-lib.org/), which is an open source library that enables technical analysis of financial markets data.
- The Java Universal Network Graph (http://jung.sourceforge.net/), which enables the analysis and visualization of graph or network based data (e.g., social networks data).

- The GeoTools (http://www.geotools.org/) toolkit, which enables the manipulation of GIS data, including the analysis of their spatial and non-spatial attributes or GIS data.
- The R project (https://www.r-project.org/), a highly extensible environment, which enables the execution of a wide variety of statistical (e.g., linear and nonlinear modelling, classical statistical tests, time-series analysis, classification, clustering) and graphical techniques.

The scope of the work that is presented in following paragraphs, involves the integration of the R project within an enhanced version of the Node-RED tool, as part of an integrated development environment offer by the VITAL smart cities platform (developed in the scope of the FP7 VITAL project). Note that the integration of IoT tools with data analytics tools is also evident in the scope of popular public cloud environment (such as the Amazon EC2 and the Microsoft Azure cloud services), which provide functionalities for IoT applications development along with data analytics toolkits.

4.3 The VITAL Architecture for IoT Analytics Applications

The VITAL IoT development environment is an integral part of the VITAL smart cities platform. This platform provide a range of tools and techniques for developing, deploying, managing and operating IoT applications in smart cities, including applications that leverage data and services from multiple IoT systems and data sources. The latter applications are based on the semantic interoperability features of the VITAL platform, which enable the repurposing and reuse of services and datasets from multiple IoT systems. An overview of the VITAL platform is provided in Figure 4.1.

The main components of the platform are:

- **Platform Provider Interface (PPI)**: The PPI is an abstract interface to underlying IoT systems and data sources, including the large number of legacy IoT systems that are nowadays available in the scope of digitally mature cities. PPI provides access to both metadata and data of the underlying systems, In particular, the information that is specified in the PPI covers system-level information, information about the internet-connected objects of the system, sensors-based observations' information (data), as well as metadata for managing SLAs (Service Level Agreements) between the operator of the VITAL platform (e.g., city authorities, telecom services providers) and the operators of the individual IoT systems.

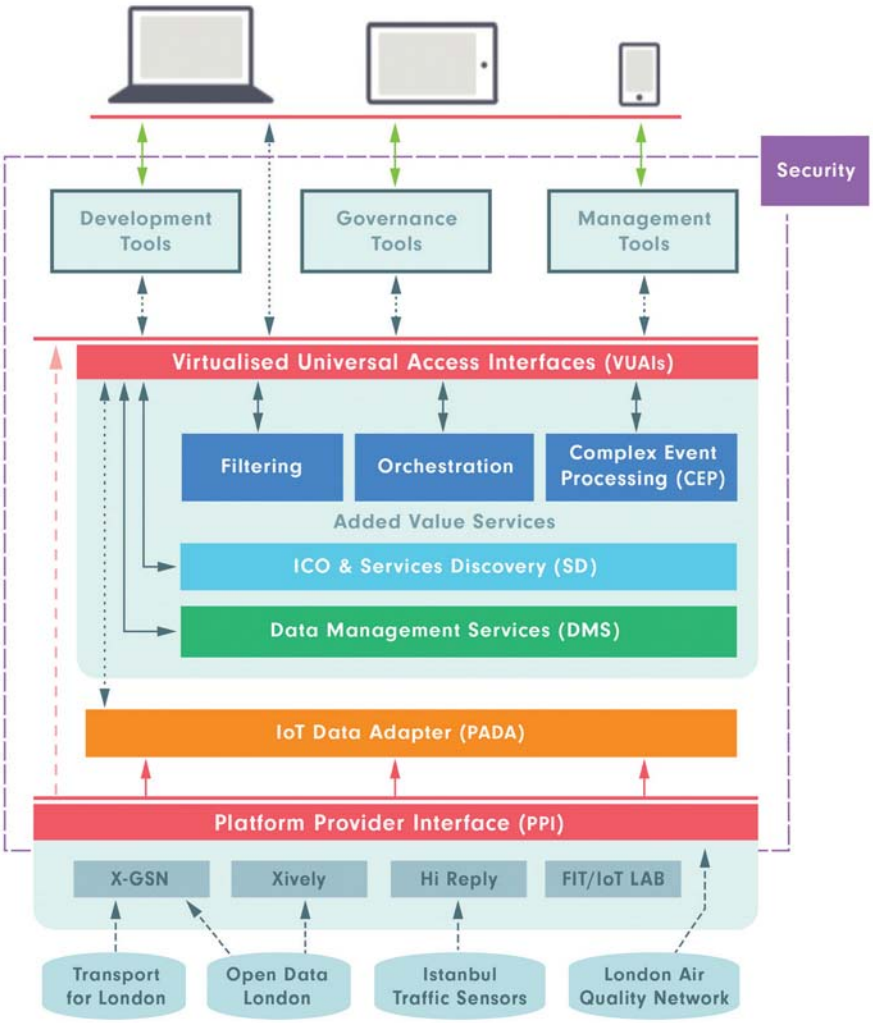

Figure 4.1 VITAL platform architecture.

- **Data Management Service (DMS)**: This is a data service (empowered by scalable operational databases), which persists and manages data from all of the underlying IoT systems. Data within the DMS are semantically unified, since they comply with the same data model (schema, ontology). Note that the DMS provides interoperable cached data from the various IoT systems, thus providing a foundation for the provision of a range of Data-as-a-Service (DaaS) services.

- **IoT Data Adapter (PADA)**: This component manages the subscriptions of the VITAL platform to IoT systems and data sources, through the management of PPIs. It therefore provides functionalities for registering and deregistering PPIs as data contributors to the DMS, while at the same time managing data acquisition from the IoT systems to the DMS (according to a publish-subscribe paradigm).
- **IoT Service Discovery (SD)**: This component enables the discovery of services, sensors, internet-connected devices and other IoT resources. In the SD context, the term "services" refers to services provided by the VITAL platform (possibly assembled based on the orchestrator component) rather than to low-level services provided by the IoT systems. The latter are typically accessible through PPIs.
- **Filtering and Complex Event Processing (CEP)**: These components offer data filtering and event generation functionalities based on data streams residing with the DMS. The filtering components support static data processing, with emphasis on threshold-based filtering and resampling. At the same time, CEP supports both static and dynamic processing of IoT streams.
- **Orchestration**: This component provides functionalities for composing workflows, thus enabling the orchestration of (composite) IoT services based on more elementary ones. As already outlined, composite IoT services produced by the orchestrator are registered to the SD component.
- **VUAIs (Virtualized Unified Access Interfaces)**: These are interfaces enabling IoT system agnostic access to data and services of the VITAL platform.

On top of the VITAL platform, three distinct environments are offered, namely:

- A management environment providing FCAPS (Fault Configuration Accounting Performance and Security) management functionalities for the VITAL modules, but also for the data and services from the underlying IoT systems.
- A governance environment enabling the configuration of the VITAL platform (including configuration of its individual modules) according to the needs and characteristics of a given urban environment. The governance environment takes into account information and parameters such as the geography and the demographics of the city in order to appropriately customize the operation of the VITAL platform.
- A development environment for producing smart city applications based on the VITAL platform. It extends the popular Node-RED tool on the

basis of functionalities for the VITAL modules, thus enabling developers to combine VITAL functionalities (e.g., orchestration, filtering, semantic interoperability) with the rich set of Node-RED functionalities. It also integrates the R project in order to boost the development of IoT Analytics applications.

Following paragraphs illustrate the VITAL development environment, as a concrete example of a tool that facilitates the development of IoT Analytics applications.

4.4 VITAL Development Environment

4.4.1 Overview

The primary goal of the VITAL development environment is to integrate all functionalities provided by the VITAL platform and make them accessible to smart city application developers through a single tool, the VITAL development tool. To this end, the various functionalities of the VITAL platform are integrated into the tool based on VUAIs, which are currently implemented as RESTful web services. This renders Node-RED ideal as the basis for the implementation of the VITAL development tool. Furthermore, the growing number of nodes (i.e. development functionalities) that are available for Node-RED, as well as its simplicity, user-friendliness, extensibility and popularity led to the selection of Node-RED as a basis for developing the VITAL tool. As shown in Figure 4.2, the VITAL development tool is based on the enhancement of Node-RED with a number of VITAL-related nodes, as well as with functionalities provided by the R project (and associated programming language for statistical computing and graphics). The result of this enhancement process is an easy-to-use tool that also enables its users to perform a number of VITAL-related (e.g. retrieval of IoT system metadata) and data analysis (e.g. data value prediction or data clustering) tasks, based on the exploitation of the VITAL platform.

A short overview of the extra nodes that have been added to the core node palette of Node-RED for the purpose of supporting and exposing the VITAL functionalities follows.

4.4.2 VITAL Nodes

In order to expose the functionalities provided by the VITAL platform through the VITAL development tool, a number of new Node-RED nodes

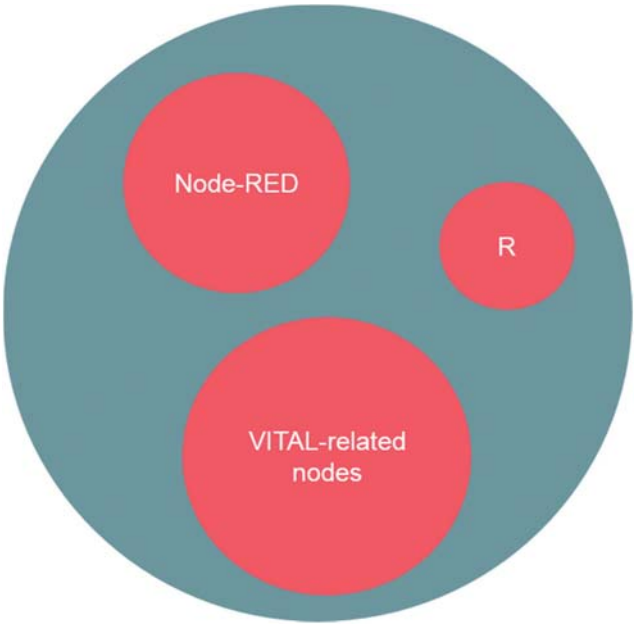

Figure 4.2 Elements of the VITAL development tool.

were created and added to the tool. More specifically, the node palette was complemented with the following node categories: (1) **ppi** that contains nodes to use in order to communicate directly with PPI-compliant IoT systems and data sources, (2) **data** that contains nodes that expose the functionalities provided by the DMS, (3) **discovery** that contains nodes that enable the discovery of different types of IoT resources, and (4) **filtering** that contains nodes that expose the filtering functionalities provided by the VITAL platform.

4.4.2.1 PPI nodes

Each node in the **ppi** category corresponds to a primitive specified as part of the Platform Provider Interface.

4.4.2.2 System nodes

System nodes are used to retrieve metadata about a PPI-compliant IoT system. When a system node receives a message, the node accesses the relevant primitive of the PPI implementation exposed by that system, and puts the result (i.e., the system metadata) into the message it finally sends out.

4.4.2.3 Services nodes

Services nodes retrieve metadata about the IoT services that a PPI-compliant IoT system provides. The message that a services node receives may contain information, which is used to filter the services to retrieve metadata for (based on their **ID** and **type**), whereas the message that a services node sends contains the retrieved service metadata.

4.4.2.4 Sensors nodes

Sensors nodes are function nodes that retrieve metadata about sensors that an IoT system manages. The messages sent to these nodes can be used to filter the sensors to retrieve metadata for (based on their **ID** and **type**), whereas the messages sent by these nodes contain the retrieved sensor metadata.

4.4.2.5 Observations nodes

Observations nodes are function nodes that pull observations made by sensors managed by a PPI-compliant IoT system. Input messages may contain information, which can be used to filter the observations to fetch (based on the **sensor** that made them, the observed **property** and the **time** when they were made), whereas output messages contain the retrieved observations.

4.4.2.6 DMS nodes

Nodes that expose functionalities provided by the DMS component of the VITAL platform fall into the **data** category.

4.4.2.7 Query systems

Query systems nodes query DMS for systems that meet specific criteria. The message that a query systems node receives contains a **query**, whereas the message that it sends out contains the metadata about all IoT systems that are registered with the VITAL platform and match the query.

4.4.2.8 Query services

Query services nodes are used to retrieve information about IoT services based on specific criteria. Input messages contain **queries**, whereas output messages contain metadata about IoT services that match those queries.

4.4.2.9 Query sensors

Query sensors nodes query DMS for internet-connected objects that meet specific criteria. The messages sent to these nodes contain a **query**, whereas the messages that these nodes send as a response contain metadata about all internet-connected objects that match the given query.

4.4.2.10 Query observations

Query observations nodes query DMS for observations. The message that a query observations node receives contains a **query**, whereas the message that a query observations node sends contains observations based on the given query.

4.4.2.11 Discovery nodes

The **discovery** node category groups together all Node-RED nodes that enable the discovery of different types of IoT resources by leveraging the discovery functionalities provided by the VITAL platform.

4.4.2.12 Discover systems nodes

Discover systems nodes are used to discover systems based on their **type** and/or **spatial context**. The messages sent to discover systems nodes contain the criteria, whereas the messages sent by these nodes contain the metadata about the systems that meet these criteria.

4.4.2.13 Discover services nodes

Discover services nodes enable the discovery of services based on specific criteria. Input messages may contain a **type** and a **system** URI, and output messages contain the available metadata about all services of that type that are provided by that system.

4.4.2.14 Discover sensors nodes

Discover sensors nodes are used to discover sensors based on their **position** (current or within a specified **time window**), **type**, **movement pattern**, **connection stability**, and whether they provide a **localizer service**. Input messages contain the criteria that sensors must meet, whereas output messages contain metadata about the sensors that meet them.

4.4.2.15 Filtering nodes

Filtering nodes are used to access the VITAL filtering functionalities.

4.4.2.16 Threshold nodes

Threshold nodes perform threshold-based filtering to the values collected from a specific internet-connected object, for a specific property, in a specific area, and within a specific time interval. Messages sent to threshold nodes contain **criteria**, based on which to retrieve observations, a **threshold** value, and a **relation**, and messages sent by these nodes contain all values that

meet the specified criteria and have the specified relation with the specified threshold.

4.4.2.17 Resample nodes

Resample nodes are used to resample (down-sample or up-sample) data streams using a different time interval than the one they were initially sampled with. Input messages specify the **data stream** (i.e., the **sensor** and the observed **property**), the new **time interval**, and the **time period**, over which to perform the resampling, whereas output messages contain the resampled observations.

4.5 Development Examples

4.5.1 Example #1: Predict the Footfall!

The purpose is to implement a web page that shows a map of Camden town. When the user clicks anywhere on that map, a pop-up appears that informs the user about the people that are expected to be walking around that area during the next hour. The expected result is shown in Figure 4.3.

In order to provide the required functionality, two flows were created using the VITAL development tool. The first flow is a web service that responds with the static HTML page that contains the Camden map. The second flow is a web service that given a location responds with a prediction for the number of people in that area within the next hour. Both flows are depicted in Figure 4.4.

The second flow receives a location, uses a **query sensors** node to find the footfall sensor that is closer to that location, uses an **observations** node to retrieve observations collected from that sensor in the last ten days, and finally leverages the **rstats** package to predict the value of that sensor in the next hour.

4.5.2 Example #2: Find a Bike!

The purpose is to build a web page that people that move in London can use in order to find out whether there are any bikes available near them. The user specifies their location on the map, and as a result a marker appears on the map for each docking station within a 500 m radius that has at least one available bike. Figure 4.6 shows the implemented web page.

Figure 4.7 depicts the two flows that were implemented for the purposes of this example. The first flow implements the web service that returns the static HTML page. The second flow receives the current location of the

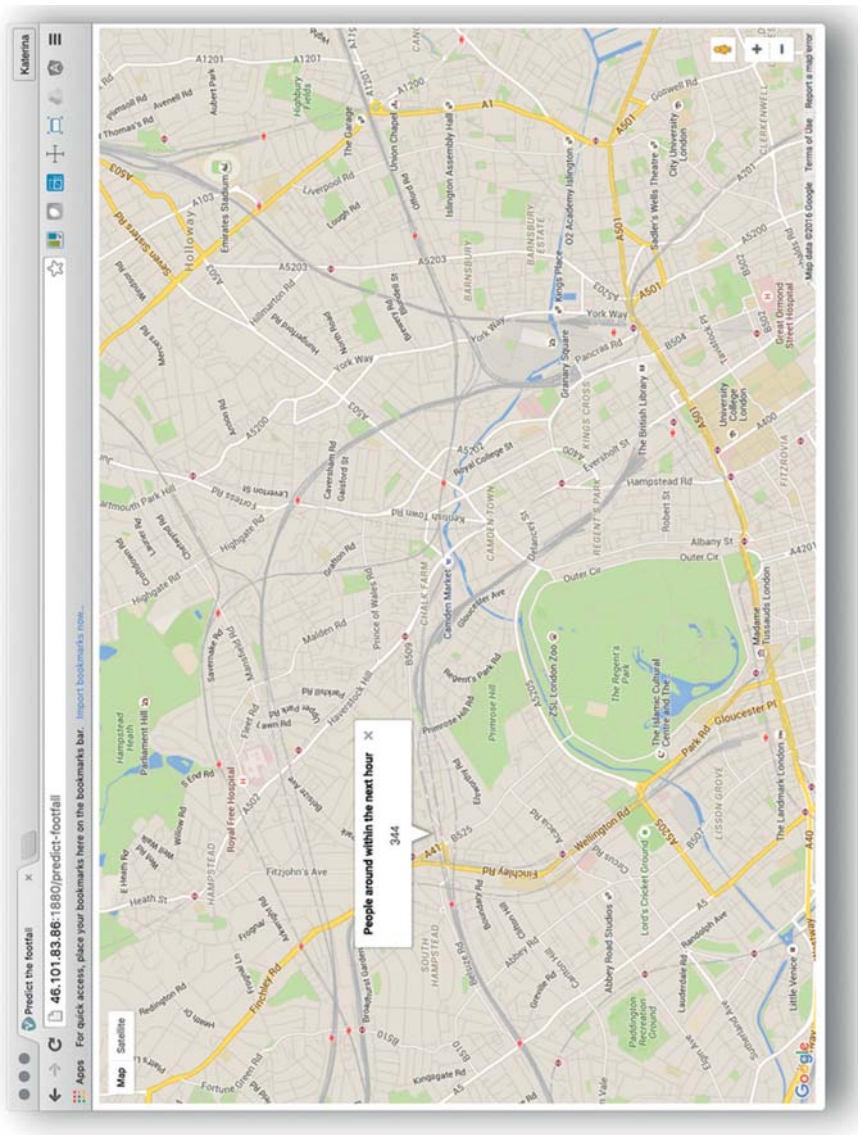

Figure 4.3 Predict the footfall – the web page.

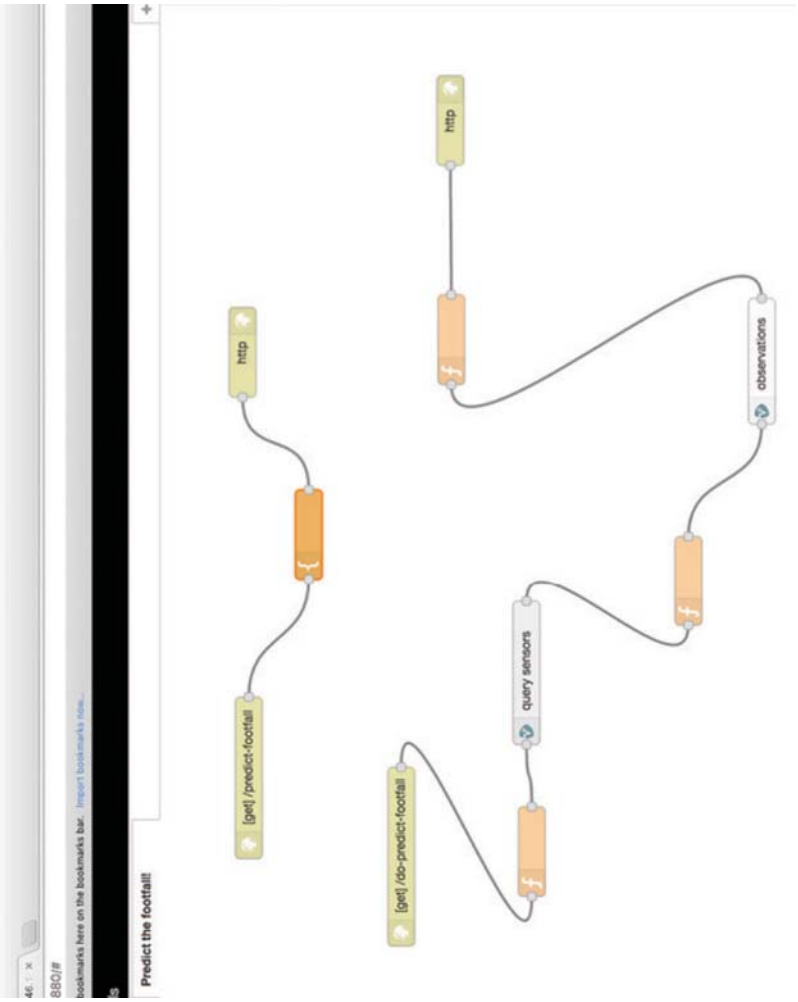

Figure 4.4 Predict the footfall – the flows.

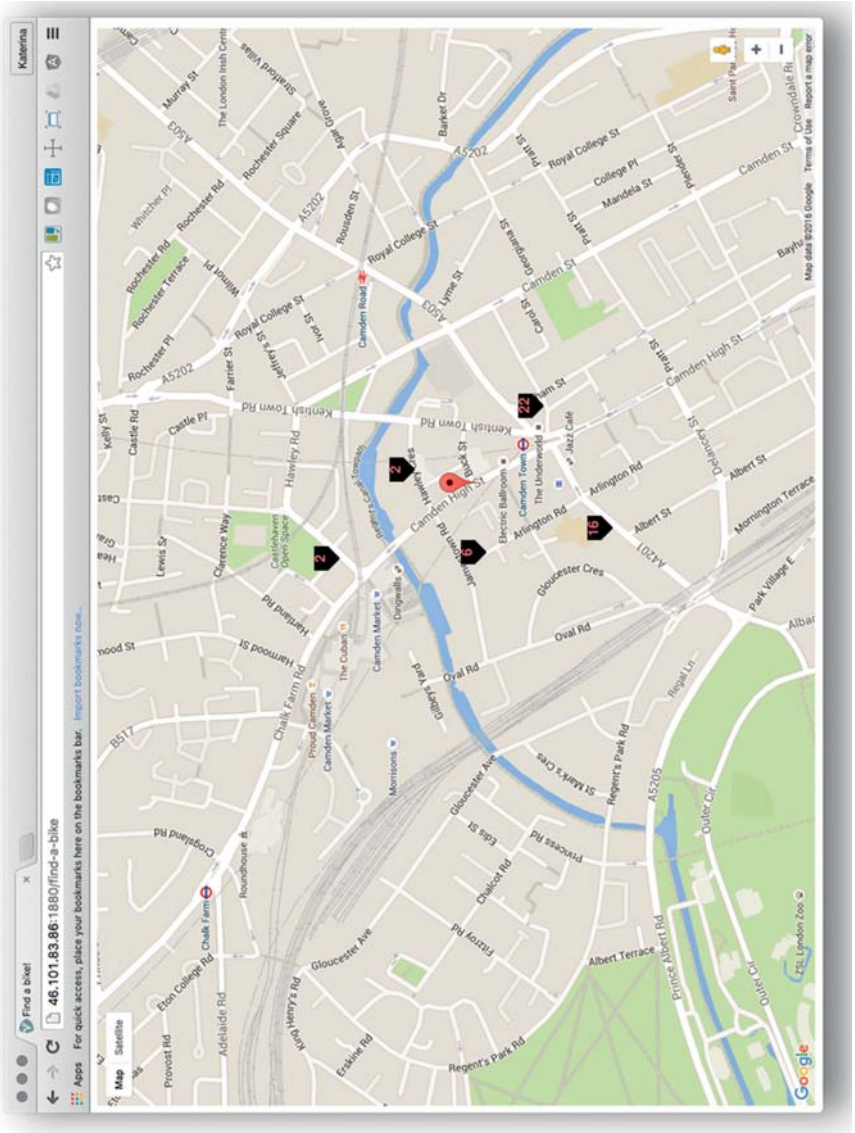

Figure 4.5 Find a bike – the web page.

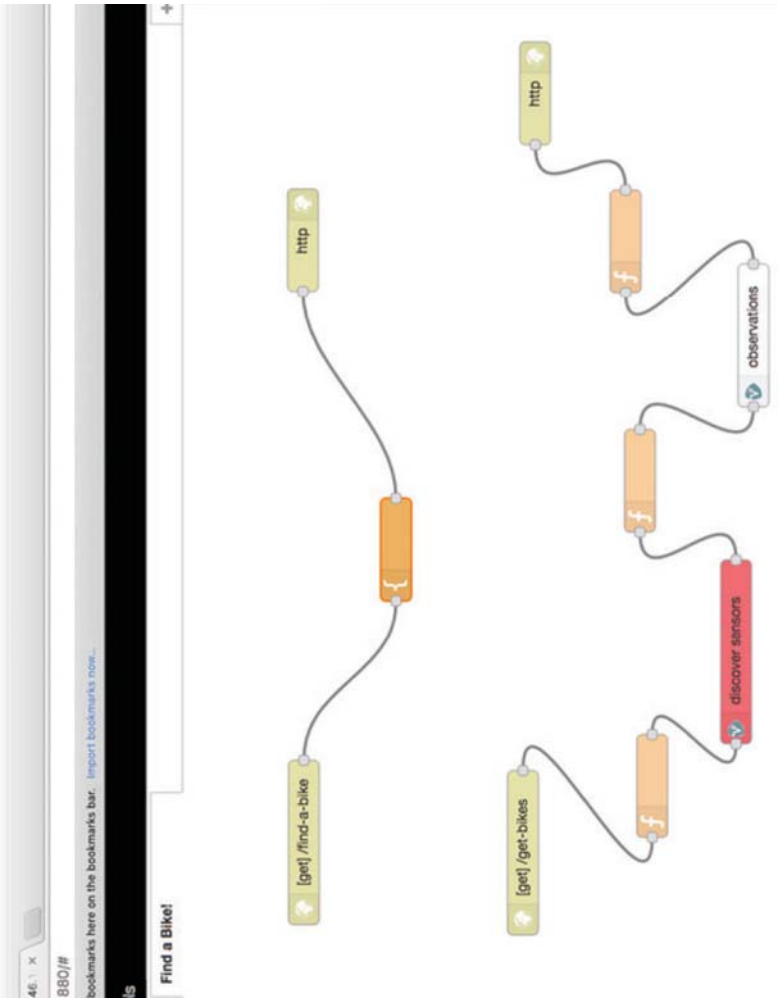

Figure 4.6 Find a bike – the flows.

user (that they have specified by clicking on the map), discovers all docking stations in that area (using a **discover sensors** node, since docking stations are essentially sensors), finds out how many available bikes each one of these stations has (using an **observations** node, since this implies the retrieval of the last observation made by each one of the corresponding sensors), and finally responds with the locations of the stations that have at least one available bike.

4.6 Conclusions

As IoT analytics applications, proliferate developers are starving for tools that can boost their development productivity. The wide array of emerging tools for IoT and data analytics applications are not enough to maximize developers' productivity, when used in isolation. Their combination and integration is therefore needed in order to achieve multiplicative benefits, i.e. leverage productivity benefits from both analytics and IoT tools. Moreover, in several cases the integration of data streaming concepts is also important, given the high velocity of IoT data streams. Integration of data streaming tools was not extensively presented in the scope of the Chapter, as VITAL stores IoT data into a scalable datastore in a semantically unified manner. However, the presented approach demonstrates also the merits of semantic interoperability for the development of added-value IoT analytics applications in smart cities, notably applications that leverage and process data from multiple IoT systems and data sources, which have typically been developed and deployed independently.

Acknowledgements

Part of this work has been carried out in the scope of the VITAL project (www.vital-iot.eu), which is co-funded by the European Commission in the scope of the FP7 framework programme (contract No. 608662).

References

[1] Karl Aberer, Manfred Hauswirth, Ali Salehi. *Infrastructure for Data Processing in Large-Scale Interconnected Sensor Networks*. MDM 2007: 198–205.

[2] Ioannis Chatzigiannakis, Georgios Mylonas, Sotiris E. Nikoletseas. *50 ways to build your application: A survey of middleware and systems for Wireless Sensor Networks*. ETFA 2007: 466–473.

[3] Achilleas Anagnostopoulos, John Soldatos, Sotiris G. Michalakos. REFiLL. *A lightweight programmable middleware platform for cost effective RFID application development*. Pervasive and Mobile Computing 5(1): 49–63 (2009).

[4] P. Patel, A. Pathak, T. Teixeira, and V. Issarny. *Towards application development for the internet of things*. In Proceedings of the 8th Middleware Doctoral Symposium, page 5. ACM, 2011.

[5] Pankesh Patel, Animesh Pathak, Damien Cassou, Valérie Issarny. *Enabling High-Level Application Development in the Internet of Things*, Lecture Notes of Sensor Systems and Software, the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering Volume 122, 2013, pp. 111–126.

[6] D. Cassou, J. Bruneau, J. Mercadal, Q. Enard, E. Balland, N. Loriant, and C. Consel. *Towards a tool-based development methodology for sense/compute/control applications*. In Proceedings of the ACM international conference companion on Object oriented programming systems languages and applications companion, pages 247–248. ACM, 2010.

[7] L. Atzori, A. Iera, and G. Morabito. *The internet of things: A survey*. Computer Networks, 54(15): 2787–2805, 2010.

[8] J. S. Brown and R.R. Burton. Diagnostic models for procedural bugs in basic mathematical skills. *Cognitive Science*, 2(2):155–192, 1978.

[9] A. Cawsey. Planning interactive explanations. I*nternational Journal of Man-Machine Studies*, in press.