Towards application development for the internet of things

Pankesh Patel, Animesh Pathak, Thiago Teixeira, Valérie Issarny

Towards Application Development for the Internet of Things

Pankesh Patel, Animesh Pathak, Thiago Teixeira, and Valérie Issarny
INRIA Paris-Rocquencourt, France
{firstname.lastname}@inria.fr

ABSTRACT

The Internet of Things (IoT) integrates the physical world with the existing Internet, and is rapidly gaining popularity, thanks to the increased adoption of smart phones and sensing devices. One of the important challenges in this domain is to enable domain experts to easily specify applications for the IoT. As a first step towards developing a suitable programming abstraction, in this paper we present a domain model for applications in the Internet of Things, based on a survey of recently proposed IoT applications from the real world that represent a wide class of behaviors found in IoT use cases.

Categories and Subject Descriptors

D.2.2 [**Software Engineering**]: Design Tools and Techniques—*Top-down programming*; D.2.11 [**Software Engineering**]: Software Architectures—*Data abstraction,Domain-specific architectures*

General Terms

Design

Keywords

Domain model, Internet of Things, Wireless Sensor Network

1. INTRODUCTION

The *Internet of Things* (IoT) has been discussed in literature for some time now, albeit with several similar but non-identical definitions. In this paper, we build upon the following definition, proposed by the CASAGRAS project [4] in 2009[1]:

"*A global network infrastructure, linking physical and virtual objects through the exploitation of data capture and communication capabilities. This infrastructure includes existing and evolving Internet and network developments. It will offer specific object-identification, sensor and connection capability as the basis for the development of independent cooperative services and applications. These will be characterised by a high degree of autonomous data capture, event transfer, network connectivity and interoperability.*"

The IoT as defined above has recently moved closer to being a reality, thanks to the increased abundance of smart phones, wireless sensor and actuator networks, and RFID technologies [1]. Several IoT applications have been reported in recent research, and we expect to see increased adoption of IoT concepts in the fields of personal health, inventory management, and domestic energy usage monitoring, among others.

An important challenge to be addressed in the domain of IoT is to enable the domain experts (health-care professionals, architects, city planners, etc.) to develop IoT applications in their fields rapidly, with minimal support needed from skilled Computer Science professionals. Similar challenges have already been addressed in the closely related fields of Wireless Sensor and Actuator Networks (WSANs) and Pervasive/Ubiquitous computing. While the main challenge in the former is the extremely *large scale* of the systems (hundreds to thousands of largely similar nodes), the primary concern in the latter has been the *heterogeneity* of nodes and the major role that the user's own interaction with these nodes plays in these systems (cf. the classic "smart home" scenario where the user interacts with a smart display which works together with his refrigerator and toaster). The upcoming field of IoT includes both WSANs as well as smart appliances, in addition to the elements of the "traditional" Internet such as Web and database servers, exposing their functionalities as Web services etc. Consequently, an ideal application development abstraction of the IoT will allow (domain expert) developers to intuitively specify the rich interactions between the extremely large number of disparate devices in the future Internet of Things.

The larger goal of this research is to propose a suitable application development framework which addresses the challenged introduced above. This will most likely be achieved by a domain specific language (DSL) which exposes the functionalities specific to the IoT to the domain experts. The first logical step towards this then is to construct a **domain model**, using the following definition from [20]:

"*A domain model is a model of the domain within which an Enterprise conducts its business.*"

We especially use the CRC – Classes, Responsibility, Collaboration technique mentioned in [20], defining the main

[1] http://www.rfidglobal.eu/userfiles/documents/CASAGRAS26022009.pdf

MDS'2011, December 12th, 2011, Lisbon, Portugal.

abstract concepts, their responsibilities, and associations that represent their relationship with each other in the IoT.

We recall from [17, 20] the following benefits of a domain model:

- **Creation of common understanding.** The different terms used by different people in the IoT domain [9] can lead to confusion, which can be alleviated by the usage of a common lexicon, as provided by a domain model. This lexicon can then be used by researchers, system programmers, as well as domain experts.
- **Modeling invariant properties.** The domain model represents the invariant properties of the domain — concepts and relationships which do not change from one application to the other. An instance of this in the IoT domain can be the notion of a *sensor* attached to a *device*. Depending on the specific applications, the type of sensors and devices can change (e.g, a light sensor attached to a smart phone), but the inherent relationship between the types of entities they represent does not.
- **Enabling modular design.** Application needs often tend to arrive in terms of behavior, which needs to be broken down and divided among the entities in the system. A good domain model aids in this process, since the capabilities of each type of entity are clearly identified. E.g., the application requirement of "the system senses the temperature of a room and keeps it steady" can be easily broken down into an application consisting of temperature sensors, computational components, and HVAC actuators, each performing its well-known role in this sense-compute-actuate loop.

In addition to the reasons above, a goal specific to the Internet of Things that a good domain model can satisfy is that of successfully integrating the models of real world things, software, and network, a need highlighted by recent works such as [14] and [16].

In this paper, we first present in Section 2 a set of representative IoT application behaviors gleaned from the latest developments in the industry and research communities. Our main contribution in this paper is a domain model (discussed in Section 3), capturing the concepts and associations needed to adequately represent the classes of IoT applications we have studied. This is followed by a discussion of the lessons learnt in Section 4. Section 5 concludes with some proposals for future work.

2. APPLICATIONS IN THE INTERNET OF THINGS

To guide us in our efforts in creating a domain model for the IoT, we first performed an extensive survey of the various applications present in the research literature as well as commercial product proposals. Our study came to the conclusion that in addition to the usual behaviors seen in the Internet — remote access to Web services, interactions by a human user with software, and database access, the Internet of Things brings the following three behaviors to the mix:

Intermittent Sensing. This behavior comes from the early definition of the IoT, which was centered around RFID technology, and is found mostly in applications where things have an *information shadow* [25] on the Internet. The reader (e.g. RFID reader, barcode reader) observes an ID of a tag and sends it to a service on the Internet, which fetches data associated with the ID from the storage and returns it to the application.

A classic case of such behavior is seen in applications such as book review sharing [11,12], meant for libraries with books that are tagged with barcodes. When a user wants to write his review about the book, he uses the application on his smart phone to scan the bar code and sends his views about the book along with its unique identifier to a storage service on the Internet. When a user wants to read the reviews of a book, he can similarly scan the barcode of the book using the same application and request the storage service for the review of the book, to which the service responds with the text of reviews gathered so far.

There are several other applications in this class, where tags can be used to carry out actions. For instance, when a child's house keys are recognized by Mir:row[2], it sends a message to the parents to say that the child has arrived home. Similarly, animals can be tagged with RFID collars[3] that can open the door to let them enter. Other instances of this class of applications are found in domains such as mobile ticketing [2], tourist information [22], etc.

Regular data collection. This behavior is seen in the class of IoT applications where (smart) things interact with users by stating information about themselves periodically. Actual objects are observed by sensors, and then the observed information is sent for various purposes (such as providing information about electricity consumption[4]) to remote users.

A novel use of this is seen in the *Botanicalls* project[5], aimed at enabling communication between plants and users. The overall functionality is that of a plant being able to notify a user on his cell phone about whether it needs water. To achieve this, a moisture sensor attached to the Botanicalls device is placed into the soil, and observes the moisture level of the soil. The Botanicalls service, which is installed on the device, translates the moisture level and sends an appropriate message. The message could be one of "water me please", "you did not water me enough", or "thanks for watering me", depending on the water level in the soil.

An evolution of this class of applications gives rise to the concept of self-logging objects that store data about themselves and their environment in great quantities. An example is the Nike+ running shoe[6], which records information such as distance and speed, and sends them to the user later. This class of applications is found in domains such as food supply chain [5, 10], patient monitoring [19], etc.

Sense-Compute-Actuate loops. This behavior is seen in applications where smart things interact with each other at either the local level or through the Internet, and provide information that can be used as new knowledge[7]. They may also take corrective actions [15] with no human originator, recipient, or intermediary, and may notify or prompt users as required.

[2] http://www.violet.net
[3] http://www.dogdoors.com/
[4] http://www.arrayent.com/
[5] http://www.botanicalls.com/
[6] http://www.apple.com/ipod/nike/run.html
[7] http://www.casaleggio.it/internet_of_things/

A simple instance of this is an application that maintains temperature in a room according to the user's preference provided using a smart phone [7]. It requests a temperature monitoring node, which hosts a temperature sensor, for temperature data. By comparing the received temperature data and user's temperature preference, the application commands a heater, which controls the temperature.

Other instances include the smart umbrella [26], which monitors the weather condition and analyzes a user's schedule, then alerts a user through a voice message when he is scheduled to go out. Glowcap[8] provides intelligence to medicine bottles using light, sound, or telephone call to remind a user when it is time to take a new dose. This class of applications is found in domains such as optimizing power consumption costs [3], work place safety[9], etc.

3. DOMAIN MODEL

Inspired by the applications discussed in the previous section, we have extracted the concepts and associations that we believe are suitable for representing applications in the Internet of Things. We present the domain model derived from them in the section below. A graphical representation of the same is in Figure 1, which follows the notation introduced in [13]. The figure contains the concepts in the IoT domain, along with their relationships with each other, including the cardinality of such relationships.

3.1 Concepts

The concepts in the Internet of Things can be divided broadly into two categories:

3.1.1 "Traditional" Internet concepts

These are concepts which developers of Internet applications are most familiar with, namely:

A **software component** (using the definition from [24]) is an architectural entity that (1) encapsulates a subset of the system's functionality and/or data, (2) restricts access to that subset via an explicitly defined interface, and (3) has explicitly defined dependencies on its required execution context. We see the following types of software components:

- A **computational service** is a software component that takes one or more units of information as input and produces an output. It is a representation of the *processing* happening in the application.
- A **storage service** provides read and write access to persistent data. This data can be accessed by other software entities by interacting with the storage service. An example would be the MySQL server providing the back-end to the book review application discussed earlier.
- An **end-user application** is a software component that is designed to help a user to perform tasks by interacting with other software components. For instance, in the room temperature maintenance application, a user can provide his temperature preferences to the IoT application using an app installed on his smart phone.

[8] `http://www.vitality.net/glowcaps.html`
[9] `http://www.sensei-project.eu/`

Some concepts related to the ones above are that of a **user**, which is a human who performs singular or multiple related specific tasks; a **store**, which is the entity that actually hosts the data (In the above example, the various databases managed by the MySQL server are instances of store); and **information**, which is any data that is meaningful by itself.

3.1.2 "Thing"-oriented concepts

In addition to the ones above, our domain model also includes the following concepts, which are speficially used to model the "things" in the IoT.

- An **entity of interest (EoI)** [8] is a real world object, including the attributes that describe it, and its state that is relevant from a user or an application perspective. For instance, the EoI may be any real world objects such as room, book, plant, etc.
- A **phenomenon** [8] a property of a physical entity that is observable. For instance, the temperature of a room and ID of tag are the examples of the phenomenon.
- A **resource** [8] is a conceptual representation of a sensor or an actuator, where a **sensor** is a type of resource that has the ability to detect changes in the environment. Thermometer, and tag readers are examples of sensors. An **actuator** is a resource that has the ability to make changes in the environment through an action. Heating or cooling elements, speakers, lights, etc. are examples of actuators.
- **Raw data** is a representation of a sensor observation. For instance, the raw data reading generated by a temperature sensor can be the number 25, without any explicit meaning or units attached to it. Note that this is different from *information* described above, which attaches additional data such as unit of measurement (Celsius/Fahrenheit). Thus, 25° Celsius is the information.
- An **action** represents the act of an entity in the environment. For instance, "switching the lights on".
- A **device** [9] is an entity that provides sensor and actuator resources the ability to communicate with other entities. Without a device, resources can not interact with other resources. Tag readers, mobile phones, and personal computers (PC) are all example devices.
- A **sensor driver** accesses raw data and further describes it by attaching metadata such as unit of measurement, time of sensing, etc. with raw data. The result is called **sensor measurement**. The sensor measurement is a type of information.
- A **command** is an instruction that describes a desired outcome. For instance, "switch the heater on", "turn on the light" are examples of commands. An **actuator driver** translates a command and triggers the actuator appropriately. For instance, the heater driver translates a command (such as "switch the heater on") and turns the heater on as a result. Note that a both sensor and actuator drivers are types of **driver**, which in turn is a type of software component (see Figure 1).

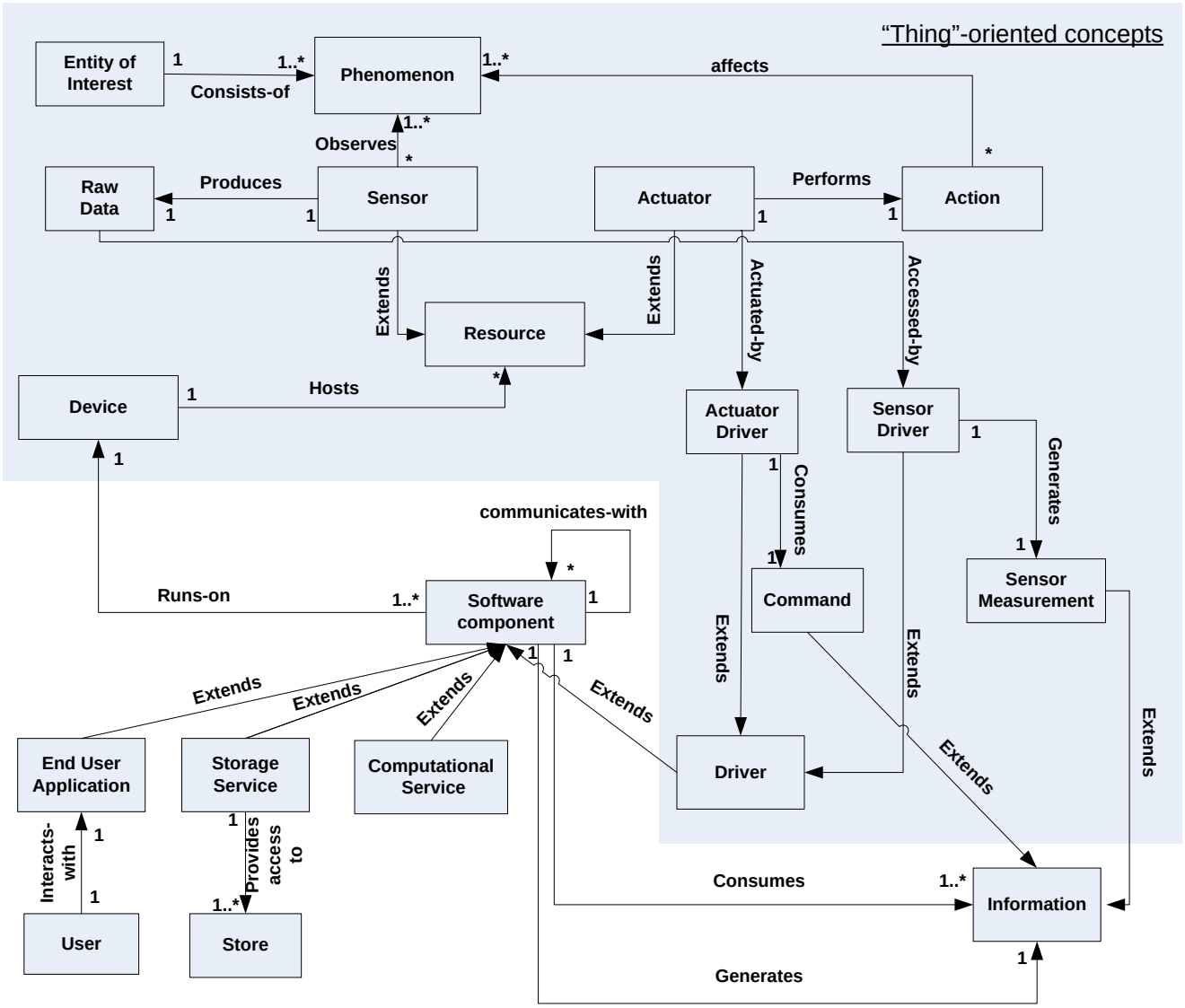

Figure 1: Domain Model

3.2 Associations

This section discusses the associations between the concepts in the Internet of Things. Where needed, we elaborate each association with the examples that are taken from the applications in Section 2. The associations are as follows:

- An EoI **consists-of** one or more phenomenon.
- A sensor **observes** a phenomenon and **produces** data about it.
- An actuator **performs** an action that **affects** a phenomenon.
- A device **hosts** zero or more resources. E.g., a smart phone might host resources such as an accelerometer sensor, a light sensor, etc.
- A software entity **runs-on** a device. E.g., an instance of the Jetty tiny web server running on a home desktop.
- Raw data is **accessed by** a sensor driver that **generates** a sensor measurement. E.g., in temperature monitoring, the temperature raw data is accessed by the temperature sensor that generates the temperature measurement.
- An actuator is **actuated by** an actuator driver that **consumes** an actuation command.
- A user **interacts-with** an end-user application for various purposes such as setting the temperature preference, reading a user's review, hearing an alert message, etc.
- A storage service **provides access to** a store. In book review sharing application (discussed in the Section 2), the storage service searches the review text in the store.
- Software components **communicate-with** each other

to exchange data and control, in a manner similar to that described under the functions of a software connector in [24]. This is one aspect of the domain model which we are actively working to further refine as part of our future work in order to identify the precise types of communications used in IoT applications, which might contain instances of various interaction paradigms [6] such as message passing, publish-subscribe etc.

4. DISCUSSION

As an initial validation of our work, we have successfully modeled three IoT applications — shared book reviews, smart plants, and room HVAC maintenance — using our domain model, representing the three behaviors discussed in Section 2. Due to lack of space, the details of those applications and their models are presented in the extended version of this text at [21].

The **core goal** of this work has been to propose a common understanding of the concepts that constitute the IoT. A clear terminology is important, not only for our own research, but also for enabling scientific discourse among researchers, system programmers, as well as domain experts [14, 16]. We note that this is not the first such effort; notably Haller [9] has defined the relationship among things, devices, resources, and services. However, this work does not mention two key concepts of IoT: phenomena and users. These two concepts are important because a user has a goal and the completion of this goal is achieved via interaction with a phenomenon of an EoI. This interaction between the user and the EoI is mediated by the IoT concepts such sensor, actuator, end-user application, sensor driver, and actuator driver.

The **second outcome** of this work is the modeling of invariant properties [20] of the IoT domain — concepts, axioms and associations which do not change from one application to other. The important advantage is that the invariant structure of the domain model makes programs more resilient to future changes. In other words, the domain model is independent of specific technologies such as RFID, barcodes, sensors, or implementations technologies such as SOAP [18] services, RESTful services [23], or Device Profile for Web Services (DPWS) [27].

The **third outcome** of this work is separation of concerns, which breaks up application requirements by dividing them among layers. In addition to the separations along "traditional" and "thing-oriented" concepts seen in Section 3, we also believe that our model can be split into three distinct layers: resource, service, and end-user. The *resource layer* includes concepts such as sensor, actuator, phenomenon, and device. The concepts in this layer are responsible for monitoring and controlling phenomena within an EoI. The *service layer* includes software components such as sensor driver, actuator driver, storage and its storage service, and computation service. These components use data generated from the resource layer and offer necessary functionality for interacting with the EoI, often by communicating with each other to accomplish a common goal. Finally, the *end-user layer* includes end-user application software components. The major advantage of this separation is that it helps to identify the capabilities of each layer.

We believe that although a lot more needs to be done in order to provide domain experts suitable application development abstractions for the IoT, this is an important, if small, first step.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we have presented, as an intial step towards a suitable application development framework for the Internet of Things, a domain model for the IoT, capturing the concepts and associations needed to adequately represent the classes of IoT applications we have studied. This is based on a survey of the applications seen in recent IoT research and industry publications.

Our future work in this area will proceed in the following directions:

Surveying the applications in the IoT. We note that although large, the current list of applications presented in this paper is not fully exhaustive. We will use this extended set of applications to further complete our view of the behaviors seen in IoT apps, which will in turn lead to an updated domain model.

Elaborating on interaction paradigms in IoT applications. Currently, our domain model deliberately does not explain the sub-classes of the "communicate-with" association between software components. This is one aspect of the domain model which we are actively working to further refine as part of our future work in order to identify the precise types of communications used in IoT applications, which might contain instances of various interaction paradigms [6] such as message passing, publish-subscribe, shared memory access etc.

Development of a high-level programming model for IoT. As stated in the introduction, our long term goal is to provide a suitable application development framework which addresses the challenges introduced above. This will most likely be achieved by a domain specific language (DSL) which exposes the functionalities specific to the IoT to the domain experts. We intend to implement an initial version of this DSL in the near future, so as to quickly validate the strength of our model.

6. ACKNOWLEDGMENTS

This work was supported in part by the European Commission FP7 CHOReOS and NESSOS projects, and the ANR Murphy project. The authors are grateful to the reviewers for their helpful comments.

7. REFERENCES

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

[2] G. Broll, E. Rukzio, M. Paolucci, M. Wagner, A. Schmidt, and H. Hußmann. Perci: Pervasive service interaction with the Internet of things. *Internet Computing, IEEE*, 13(6):74–81, 2009.

[3] C. Buckl, S. Sommer, A. Scholz, A. Knoll, A. Kemper, J. Heuer, and A. Schmitt. Services to the field: An approach for resource constrained sensor/actor networks. In *2009 International Conference on Advanced Information Networking and Applications Workshops*, pages 476–481. IEEE, 2009.

[4] CASAGRAS EU project final report. `http://www.rfidglobal.eu/userfiles/documents/FinalReport.pdf`.

[5] A. Dada and F. Thiesse. Sensor applications in the supply chain: The example of quality-based issuing of perishables. In *Proceedings of the 1st international conference on The internet of things*, pages 140–154. Springer-Verlag, 2008.

[6] P. Eugster, P. Felber, R. Guerraoui, and A. Kermarrec. The many faces of publish/subscribe. *ACM Computing Surveys (CSUR)*, 35(2):114–131, 2003.

[7] M. Feldmeier and J. Paradiso. Personalized HVAC control system. In *Internet of Things (IOT), 2010*, pages 1 –8, 29 2010-dec. 1 2010.

[8] A. Gluhak, M. Bauer, F. Montagut, V. Stirbu, M. Johansson, and M. Presser. Towards an Architecture for the Real World Internet. *Towards the Future Internet*, page 313, 2009.

[9] S. Haller. The Things in the Internet of Things. *Poster at the (IoT 2010). Tokyo, Japan, November*, 2010.

[10] A. Ilic, T. Staake, and E. Fleisch. Using Sensor Information to Reduce the Carbon Footprint of Perishable Goods. *Pervasive Computing, IEEE*, 8(1):22 –29, jan.-march 2009.

[11] A. Jara, A. Alcolea, M. Zamora, A. Skarmeta, and M. Alsaedy. Drugs interaction checker based on IoT. In *Internet of Things (IOT), 2010*, pages 1–8. IEEE, 2010.

[12] S. Karpischek and F. Michahelles. my2cents-Digitizing consumer opinions and comments about retail products. In *Internet of Things (IOT), 2010*, pages 1–7. IEEE, 2010.

[13] C. Larman. *Applying UML and patterns: an introduction to object-oriented analysis and design and the unified process*. Prentice Hall, 2002.

[14] Y. Liu. Toward a unified object model for cyber-physical systems. In *Proceeding of the 2nd workshop on Software engineering for sensor network applications*, pages 65–66. ACM, 2011.

[15] F. Mattern and C. Floerkemeier. From the Internet of Computers to the Internet of Things. *From active data management to event-based systems and more*, pages 242–259, 2010.

[16] S. Meyer, K. Sterner, C. Magerkurth, J. Pasquier, D. Guinard, M. Muller, A. Abdaladhem, A. Albreshne, P. Fuhrer, J. Pasquier-Rocha, et al. Towards modeling real-world aware business processes. In *Proceedings of the Second International Workshop on Web of Things*, page 8. ACM, 2011.

[17] G. Muller. A reference architecture primer. *Eindhoven Univ. of Techn., Eindhoven, White paper*, 2008.

[18] E. Newcomer. *Understanding Web Services: XML, Wsdl, Soap, and UDDI*. Addison-Wesley Professional, 2002.

[19] D. Niyato, E. Hossain, and S. Camorlinga. Remote patient monitoring service using heterogeneous wireless access networks: architecture and optimization. *Selected Areas in Communications, IEEE Journal on*, 27(4):412 –423, may 2009.

[20] P. Oldfield. Domain Modelling. Technical report, Appropriate Process Group, 2002.

[21] P. Patel, A. Pathak, T. Teixeira, and V. Issarny. Modeling applications for the internet of things. `http://www-rocq.inria.fr/who/Animesh.Pathak/papers/iotmodel-report.pdf`.

[22] D. Reilly, M. Welsman-Dinelle, C. Bate, and K. Inkpen. Just point and click?: using handhelds to interact with paper maps. In *Proceedings of the 7th international conference on Human computer interaction with mobile devices & services*, pages 239–242. ACM, 2005.

[23] L. Richardson and S. Ruby. *RESTful web services*. O'Reilly Media, 2007.

[24] R. Taylor, N. Medvidovic, and E. Dashofy. *Software architecture: foundations, theory, and practice*. Wiley, 2009.

[25] C. A. Valhouli. The Internet of Things: Networked Objects and Smart Devices. Technical report, the hammersmithgroup, February 2011.

[26] J. Vazquez and D. Lopez-de Ipina. Social devices: autonomous artifacts that communicate on the Internet. In *The Internet of things: first international conference, IOT 2008, Zurich, Switzerland, March 26-28, 2008: proceedings*, volume 4952, page 308. Springer-Verlag New York Inc, 2008.

[27] E. Zeeb, A. Bobek, H. Bohn, S. Prüter, A. Pohl, H. Krumm, I. Lück, F. Golatowski, and D. Timmermann. Ws4d: Soa-toolkits making embedded systems ready for web services. *Open Source Software and Productlines 2007 (OSSPL07)*.