Web of Things Augmentation

Jose Lobo
LIFIA, Fac. Informática, UNLP
Argentina
jloboprs@gmail.com

Sergio Firmenich
LIFIA, Fac. Informática, UNLP and CONICET
Argentina
sfirmenich@lifia.info.unlp.edu.ar

Gustavo Rossi
LIFIA, Fac. Informática, UNLP and CONICET
Argentina
gustavo@lifia.info.unlp.edu.ar

Nahuel Defossé
DIT, LINVI, Fac. Ingeniería, UNLPSJB
Argentina
ndefosse@tw.unp.edu.ar

Manuel Wimmer
BIG, TU Wien
Austria
wimmer@big.tuwien.ac.at

ABSTRACT

The importance of the Internet of Things (IoT) in our society is reflected in its exponential growth over recent and years to come. The Web of Things (WoT) emerged as a special case of IoT, allowing end-users to deal with their devices through Web applications, with which they are really familiarized. However, as more users are reached by this technology it becomes more difficult to fulfill specific user needs, i.e. user's requirements these technologies are not prepared to cover. For that matter this paper presents an approach that proposes augmenting smart devices for the WoT through the augmentation of their corresponding Web applications. In this way, devices may be enriched with new behavior (composed by the existing ones) to better fit further user needs. We propose Domoto as a solution that achieves device augmentation, composed by a framework to build browser extensions that augments devices behavior, and a Web Application that manages devices and extensions.

KEYWORDS

Web of Things, Web Augmentation, End-user requirements

1 INTRODUCTION

The WoT was proposed as a specialization over IoT, focusing on the application layer rather than network protocols [1], using open services that employ the well-known web standards [2]. In this way, compatibility problems among devices are avoided, allowing to use popular languages and technologies on the WEB (JavaScript, Python, REST, HTML, mashup tools, etc.) [2][4].

The IoT stops being a set of isolated devices and sensors, to become into a set of open-services-oriented devices able to communicate with totally different systems thanks to the WoT [5]. In this way, on the one hand, several existing works propose user-independent scenarios where devices may be synchronized and put to work altogether automatically [9]. Nevertheless, on the one other hand, this *webification* ends in the fact that end-users may deal with their devices through native apps (for instance controlling their devices from a mobile phone) or through Web applications, that allow users to control their devices from any Web browser. This has a very good first consequence, as it is pointed in existing literature [3], that the learning curve is reduced not only for developers but also for end-users, who already got used to deal with very complex Web applications. A second consequence of using Web applications for the WoT is that these original applications may not fulfill user's requirements. Nevertheless, to *modify* externally third-party Web sites to satisfy non contemplated user requirements, i.e. adapting existing Web sites with new content and functionality, is being studied from some years ago, mostly through the Web Augmentation technique [8].

In this paper we investigate how smart devices may be augmented in order to improve the users' direct interaction with them, taking advantage of their original behavior to create new ones that augment what is possible to do with them.

Imagine a user who has a smart stove, with simple operations such as turn on, turn off, set temperature, and get status. In this scenario, the user may want further behavior, such as programmable turn on, the execution of recipes (that changes temperatures and state in small routines), etc. In some way, these are user's requirements (related to the interaction with the smart device) that are not satisfied. We present an approach for tackling this behavioral gap by augmenting smart devices firstly, which is a previous and empowering step for future integrations with other devices where more complex behavior may be reused. For making the augmentation concretely perceivable by end-users, we propose a framework that, besides allowing developers to create the augmentation they want for a given device, builds the necessary UI components to allow end-users to trigger this new behavior.

2 RELATED WORKS

Although the focus of this paper is on the WoT, we base our contribution in the technique called Web Augmentation. The

Augmented Web, as it was recently coined, "is to the web what augmented reality is to the physical world: to layer relevant content/layout/navigation over the existing web to improve the user experience" [8]. From our point of view, the Web platform growing around the WoT reaches the same condition: there are unsatisfied user requirements related to the use of existing WoT applications. Big communities have emerged (such as greasefork.org and userstyles.org), where end-users and programmers collaborate in the use, creation and maintenance of these software artifacts for adapting third-party Web sites [10].

The combination of the power of Web augmentation and the high user acceptance that Web augmentation artefacts have, makes this technique very attractive to be implemented in WoT. As it is pointed in existing literature [13], the WoT may require further enablers to reach a wider user base. With this in mind, the authors present WoTkit, a toolkit for empowering developers to create WoT applications in a simpler way. Is worth noting that other frameworks for developing WoT applications have emerged too, like WebPlug, which is mainly oriented to device integration [14]. Perhaps one of the more interesting examples is IFTTT, a Web application that allows end-users to *put the internet to work for them* by specifying simple IFTTT applets. These applets are based on the IF/THEN conditional structure. With just some clicks plus simple configuration, end-users can turn the lights on as when they arrive to their homes[1] or get an email with the location they have parked the car[2].

IFTTT and other similar solutions for physical mash-ups, introduced in the following section, allow users to integrate existing WoT devices, but to our knowledge, they still lack augmentation of their applications, which could improve users experience when they are interacting directly with a particular device. This kind of integration is inherent to WoT, thought to make devices interoperable, but several aspects are inherited from web mashups, a well-known technique for composition and reuse of both content and services. This integration is more interesting when a good number of intelligent devices are required to be managed, especially when a great deal of composition and communication between devices is essential, resulting in a completely new application. The WoT community has defined this technique as physical mashups, mainly for device management, treating each element as either a data source or an interface that can be easily integrated into the Web [6][2]. Tools such as Node-RED allow to integrate different devices and data sources treated as blocks that can be connected to each other to compose a flow and where user defined software routines can also be included as blocks [7].

During recent years the WoT community has worked in this area focused on integration between different devices which led to adaptation to some degree. Although Physical Mashups allow smart device integration and boasts a wide thriving community, they are mainly focused on composition rather than adaptation. However, it shows that end-users need further software for managing their devices. One of the best examples of this is aforementioned IFTTT. There are three thousands IFTTT applets, that were defined by end-users without programming skills and shared in the community.

From our point of view, although these approaches allow and improve the integrability of smart devices, they do not take into account the direct interaction between the end-user and a specific device. Similar to Web mashups, where the users finally interact with a totally new application (the resulting mashup) instead of interacting individually with each component, physical mashups also propose to generate applications (or interaction flows) that do not allow end-users to control directly one of the integrated devices. This direct management is something quite relevant for tackling opportunistic requirements in the use of smart devices. Different to current works in the literature, in this paper we do not tackle the integration problem as one of the main goals, but we focus our contribution around the interaction between users and devices. With the objective of improving users' experience managing a single device first, then augmenting these devices with behavior that originally is not supported and finally easing their operation. However, we believe that devices integration is very important, and consequently our approach is open to integrate all the device behaviors (both the original and augmented ones).

3 THE APPROACH IN A NUTSHELL

Our idea is depicted in Figure 1. At the left, we present the overview of the architecture, the core of the Domoto approach. The main idea is that, given a particular smart device (which exposes its services through a Web API, a Web application or both), a specification is defined. This specification represents the device behaviors, and has the responsibility of keeping a communication with it. On top of this specification, new behavior is added, which is either based on the original behaviors or added by augmentation. Both the device specification and the augmentation are developed using our augmentation framework, called Domoto.

It is worth noting that this framework in its self is independent of the execution context, it could work from inside a Web browser extension (allowing the augmentation of original smart devices' Web clients) but also from the DomotoUI application (presented in Section 4), that allows end-users to interact with all their devices.

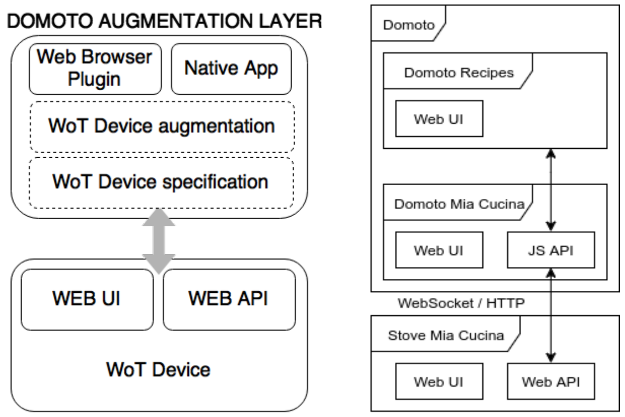
Continuing the previous example, the Figure 1.b shows a recipe augmentation for the smart stove. Basically, it uses the original device's API, on top of which new behavior is defined and exposed through the DomotoUI application.

Summarizing, Domoto approach is composed by:

- Domoto Framework: which provides the software architecture for augmenting smart devices with new behavior.
- DomotoUI application: which offers a context for executing Domoto augmentations from a native application (in order to support devices that do not offer native Web clients).

[1] https://ifttt.com/applets/TXB65wWg-automatically-turn-on-your-hue-lights-when-you-unlock-your-lockitron
[2] https://ifttt.com/applets/346212p-automatically-get-an-email-every-time-you-park-your-bmw-with-a-map-to-where-you-re-parked

- Domoto Web Extension: which allows the execution of Domoto augmentations from inside the Web browser (by augmenting the original Web applications provided by their devices).



(a) Architecture overview (b) Example of concrete device augmentation

Figure 1: DOMOTO approach

3.1 Domoto Framework

The Domoto framework has two main responsibilities:
- Allows developers to define easily the device specification.
- Provides a simpler way for developing new behaviors given the original ones.
- Builds the corresponding UI for the new behavior, in order to allow users to execute it.

An excerpt of the design of the Domoto framework is shown in Figure 2. Domoto framework is instantiated by creating a subclass of the Domoto class. For a new Domoto extension, several aspects related to the UI are available in the framework, reducing the effort required from developers point of view. For the sake of space, we do not focus on this UI generation aspect but in the part of the framework related to the device management and augmentation, which is addressed with the Module class that represents a device.

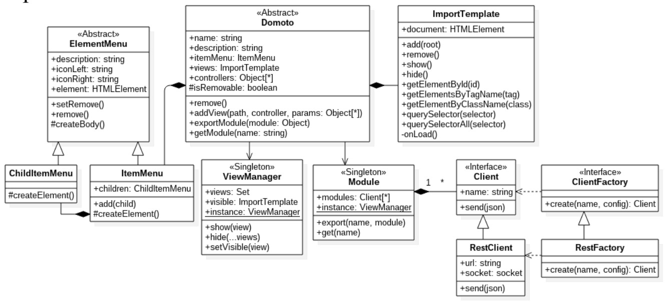


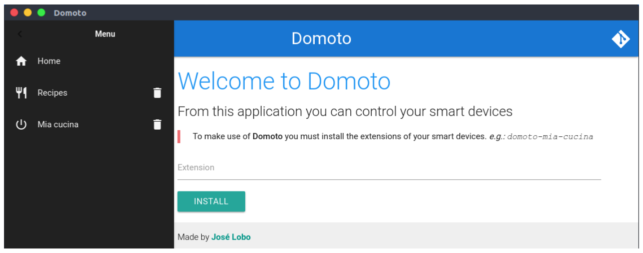
Figure 2: DOMOTO Framework

To allow other extensions to add or change the behavior of smart devices, it is necessary to share an object using the method *exportModule* in Domoto, so that other extensions can obtain it using the *getModule* function with the name of the extension. Thus the shared object can be modified and used to add behavior that was not initially implemented by the developers.

3.2 DomotoUI: an end-user application

To achieve the device management, we built the DomotoGUI application, which is responsible for managing the smart devices through the installed plugins and allows end-users to interact with their devices using both their original and augmented behavior. This was built with Electron, a tool that allows the construction of native applications with Web technologies. Thus the techniques applied in Web Augmentation can be implemented in desktop applications.

Device linking is done by installing the extensions in the application, the extensions are responsible for both the connection to the devices as well as their configuration.



Welcome to Domoto

From this application you can control your smart devices

To make use of **Domoto** you must install the extensions of your smart devices. *e.g.:* `domoto-mia-cucina`

Made by **José Lobo**

Figure 3: DomotoUI Application

3.3 Scenario of use

As a case study we propose the construction of an intelligent stove (Mia Cucina), which emulates a product available in the market. Web augmentation techniques are applied to this device in order to provide the ability of cooking with programmed recipes, on top of the manufacturer provided turn on and off functionality.

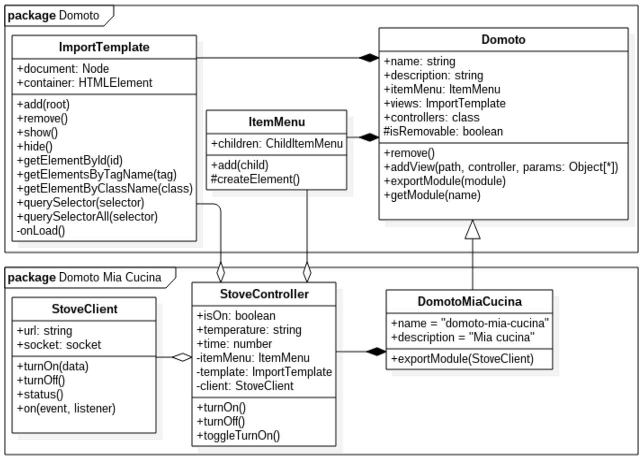


Figure 4: Domoto Mia Cucina

The stove has a hardware component which controls its states (on and off), as well as the temperature (high, medium and low). It also has a software component, which provides a user facing Web interface, and a REST API for further interaction with other systems. The hardware consists of a Raspberry board interacting with the electrical resistances of the stove, providing ignition and temperature control. The original REST API for the stove has two endpoints: turnOn (that accepts temperature and duration as parameters) and turnOff. These are responsible for turning the stove on and off respectively. This is represented by the class StoveClient, in Figure 4.

The stove has an initial Web application, on top of which we built the Domoto extension. The extension can be distributed by the manufacturer to allow the augmentation of their device, but when this is not the case, the Web developer community can create the extension using the services that stove already has. The extension shares the object that manages the stove so that other extensions can also control the device and add behavior.

The extension augments the behavior of the stove allowing cooking with recipes, for instance a recipe for cooking rice. This recipe uses a wizard that makes use of sequence of on, off and temperature change commands. These instructions are sent to the stove API using the object that was shared by the Domoto Mia Cucina extension, while at the same time indications are shown to the user with on screen messages. The new Domoto applications may have a UI counterpart that is easily created by using the Domoto framework (as shown in Figure 4). There are two ways of creating this visual control. First, these may be used from the DomotoUI application, where the sidebar is populated with all the installed extensions, among them the "recipes" extension appears, as shown in Figure 3. The second way is augmenting the original Web site, if this is desired, through the Domoto Web extension.

4 CONCLUSIONS

This document proposes the application of web augmentation on intelligent devices for the Web of things, as a mechanism that achieves device adaptability to cover a wider range of user needs. We also present an implementation of the framework as a case study for the augmentation of a smart stove, to which cook functionality is added through a native desktop Web platform. This platform allows adding or extending augmentations developed by the user's community.

ACKNOWLEDGMENTS

This work was partially supported by MINCYT Argentina, Project PICT PICT-2015- 2050.

REFERENCES

1. D. D. Guinard and V. M. Trifa, Building the web of things, Manning Publications, 2016, p. 344.
2. D. Guinard, V. Trifa and E. Wilde, "A resource oriented architecture for the web of things," in 2010 Internet of Things (IOT), IEEE, 2010, pp. 1-8.
3. C. Pautasso, O. Zimmermann and F. Leymann, "Restful Web Services vs. "Big"' Web Services: Making the Right Architectural Decision," in Proceedings of the 17th international conference on World Wide Web, New York, ACM, 2008, pp. 805-814.
4. D. Guinard, V. Trifa, F. Mattern and E. Wilde, "From the internet of things to the web of things: Resource-oriented architecture and best practices," in Architecting the Internet of Things, Springer, 2011, pp. 97-129..
5. D. Guinard and V. Trifa, "Towards the Web of Things: Web Mashups for Embedded Devices," in Workshop on Mashups, Enterprise Mashups and Lightweight Composition on the Web (MEM 2009), in proceedings of WWW (International World Wide Web Conferences), Madrid, 2009, p. 15.
6. R. Kleinfeld, S. Steglich, L. Radziwonowicz and C. Doukas, "glue. things: a Mashup Platform for wiring the Internet of Things with the Internet of Services," in Proceedings of the 5th International Workshop on Web of Things, New York, ACM, 2014, pp. 16-21.
7. JS Foundation, "Node-RED," [Online]. Available: nodered.org. [Accessed September 2017].
8. O. Díaz and C. Arellano, "The augmented web: rationales, opportunities, and challenges on browser-side transcoding," ACM Trans. Web, vol. 9, no. 2, p. 8, 2015.
9. Boussard, M., Christophe, B., Le Berre, O., & Toubiana, V. (2011, June). Providing user support in web-of-things enabled smart spaces. In Proceedings of the Second International Workshop on Web of Things (p. 11). ACM.
10. Firmenich, D., Firmenich, S., Rivero, J. M., Antonelli, L., & Rossi, G. (2016). CrowdMock: an approach for defining and evolving web augmentation requirements. Requirements Engineering, 1-29.
11. Bigham, J. P., & Ladner, R. E. (2007, May). Accessmonkey: a collaborative scripting framework for web users and developers. In Proceedings of the 2007 international cross-disciplinary conference on Web accessibility (W4A) (pp. 25-34). ACM.
12. Díaz, O., Aldalur, I., Arellano, C., Medina, H., & Firmenich, S. (2016). Web mashups with WebMakeup. In Rapid Mashup Development Tools (pp. 82-97). Springer, Cham.
13. Blackstock, M., & Lea, R. (2012, October). IoT mashups with the WoTKit. In Internet of Things (IOT), 2012 3rd International Conference on the (pp. 159-166). IEEE.
14. Ostermaier, B., Schlup, F., & Römer, K. (2010, March). Webplug: A framework for the web of things. In Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010 8th IEEE International Conference on (pp. 690-695). IEEE.