The Direwolf Inside You: End User Development for Heterogeneous Web of Things Appliances

István Koren(✉) and Ralf Klamma

Advanced Community Information Systems (ACIS) Group,
RWTH Aachen University, Ahornstr. 55, 52056 Aachen, Germany
{koren,klamma}@dbis.rwth-aachen.de
http://dbis.rwth-aachen.de

Abstract. Mobile computing devices like smartphones have become a commodity. They are very convenient when connecting to ubiquitous Web of Things (WoT) appliances. However, WoT manufacturers are challenged to provide Web application interfaces for a multitude of mobile platforms in a short time. Moreover, end users are required to install dedicated Web apps for giving them access to these emerging technologies. To overcome this situational overburdening efforts, end user development in the form of component-based Web mashups has already been applied successfully in various domains. In this paper, we envision a framework for letting users create situational applications for opportunistic device usage. We explore the recent Web Component group of W3C recommendations as a foundation for peer-to-peer cross-platform, cross-application and cross-user Web applications. Our preliminary experiences may help the Web engineering community to build better Web infrastructures for a heterogeneous device landscape.

Keywords: Web of Things · End User Development · Web Components

1 Introduction

Over the last years, we have seen a massive growth in the number of smartphones used. They enable us to instantly access any kind of information while on the go; millions of platform-specific, native apps are serving individual needs like games, news, office and social networking applications. More recently, we encounter an exponentially growing number of everyday devices connected to the Internet: online TVs, intelligent power sockets, and many other ambient technologies form the basis of the *Internet of Things*, or *Web of Things (WoT)* in the context of the Web. Already, we are confronted with a multitude of WoT appliances, from interactive hotel rooms to conference venue systems that can be controlled from mobile devices. However, they often either require using provided hardware or installing a dedicated app on a personal mobile computer. Both options come with difficulties: We either have to learn how to use the provided input modalities, or we need to go to an app store to download an app. Thus, the time-consuming access methods of these WoT resources are not

A. Bozzon et al. (Eds.): ICWE 2016, LNCS 9671, pp. 484–491, 2016.
DOI: 10.1007/978-3-319-38791-8_35

adapted to the short-term situational context they are used in. Additionally, many companies producing these appliances are from industries not traditionally linked with the IT and software world; they are confronted with enormous economical challenges to provide user interfaces to a wide variety of end user smartphones and wearables. Yet, what unites both the WoT world and end user mobile tools is the universal access to the Internet, and in particular the standardized World Wide Web with browsers accessing HTML5 resources served over the HTTP protocol.

Software engineering traditionally strives to create development artifacts that are maintainable and reusable across projects. In the Web context, cleanly-cut responsibilities and functionalities are visible in the agile development of microservices and in the composition of Web 2.0 mashups [1]. These standards-based HTML5 mashups are ideal platforms for various situational use cases [2]. In this regard, *End User Development (EUD)* is commonly defined as a methodology to allow users of software systems to act as non-professional developers to create, modify or extend software artifacts [3].

In this article, we explore the research question, what infrastructure is needed for component-based Web engineering practices to unite the scattered world of mobile (smartphone and wearable) apps and everyday devices connected to the Web of Things. We embed our research in the field of End User Development in Sect. 2 and present related work. Section 3 discusses the extension of DireWolf, a platform for multi- and cross-device Web user interfaces. By including user interface elements referenced or served directly by WoT devices, our framework acknowledges the heterogeneity and fast evolution of device-specific proprietary APIs. Section 4 highlights implementational aspects building on the recent Web Components group of W3C recommendations[1]. Section 5 concludes the article by presenting a preliminary evaluation and giving an outlook on future work. We are confident, that using standardized HTML5 components will in the long run ease the development of user interfaces for various device types while at the same time liberating users from the tedious task of having to install separate apps for interacting with diverse everyday objects.

2 End User Development for the Web of Things

In traditional software engineering, usually developers are creating software to be used by users. The idea of End User Development is to break with this classical roles and give more power to the users to design their application. It is based on the idea, that end users know the best about their surroundings - the context, the tools and their constraints. Spreadsheets are generally considered as the first broad emergence of the EUD paradigm [4], in that users are creating formulas that resemble algorithms to calculate values based on fields. Since the early findings, a broad range of research has been carried out in the field, up to economical topics [5,6].

[1] https://www.w3.org/standards/techs/components.

Situational apps created for short-term needs in highly specialized environments are ideal candidates for End User Development [2,3]. In that sense, they resemble the characteristics of the *Long Tail* [7]. Originally conceived in the area of e-commerce, standing for the idea that the bulk of sales is not in the few top items but in the rest, the term is now also used for the large number of niche communities with specialized needs. Especially composite Web applications are now being associated with the Long Tail [8], as they allow context-dependent situational usage.

2.1 Related Work for the Web of Things

The world of everyday objects connected to the Internet is scattered with vendor- and device specific apps. On the protocol level, there exists a bewildering variety of standards such as XMPP, MQTT, CoAP and proprietary pseudo-standards like Z-Wave and EnOcean. The Web of Things by Guinard et al. [9] stands for the idea that every resource in the Internet of Things is accessible over the HTTP protocol, either directly or over gateways. In the WoT concept, the devices provide both, a JSON representation and a basic HTML interface [10], however advanced user interface concepts are not in the prime scope of the articles.

More recently, the Physical Web[2] approach by Google is based on the concept that devices broadcast a URL via Bluetooth that can be read out by users' mobile devices, pointing to a cloud-based application able to interact with the device through WebSockets. While this concept allows to access WoT devices through arbitrary Web-capable clients, it does not cater for building coherent applications out of composite parts; i.e. for controlling dozens of devices in a smart home, various bookmarks need to be kept around.

Snap-To-It is a platform for opportunistic discovery of devices based on photographs [11]. The authors performed user studies to find out the preferred way of consumers to interact with on- and offline devices in everyday situations. Neither QR codes nor list-based approaches were the preferred method of interaction; instead, photographs of hardware, software and physical artifacts like maps were favored and later implemented. To support the system, additional computing-intensive resources are needed in the network for the image discovery. While our work would benefit of the advanced object recognition capabilities to make connecting to objects easier, we take the coupling for granted and focus on the composability on the user interface level.

Multiple approaches for component-based Web applications have emerged over the last years in different domains, for instance the OMELETTE platform for telco mashups [12] and the ROLE SDK for personalized learning environments [13]. Special requirements arise as soon as mobile devices are included in these mashups. While advances have been made in the overlapping area of *Distributed User Interfaces*, another topic that profits by componentized interfaces, existing work has focused on native applications [14,15] and/or desktop-based

[2] https://google.github.io/physical-web/.

composition paradigms [15,16]. What is missing, is a framework that spans various types of Web-capable devices. In the next section, we discuss our vision towards device-agnostic componentized Web interfaces.

3 User Interfaces for the Web of Things

In our conceptual framework, we combine ideas of End User Development, the Web of Things and the Physical Web and allow devices to broadcast their own user interface and access logic. We are able to read in the user interfaces and display them in a common Web platform. To this end, we extend DireWolf, a framework for multi-device widget-based Web applications [17]. DireWolf already gives us the conceptual notion of sharing Web interfaces by synchronizing Web applications across multiple devices.

Figure 1 shows a system overview of the extended DireWolf framework in a smart ambient setting with intelligent lighting and a weather station connected to the Internet. The approach unites various flavors of end user devices with local and cloud-based solutions accessing the Web of Things. Smart things advertise an URL either by QR codes or Bluetooth Low Energy beacon signals. Either way, the URL points to a user interface element resource downloadable through HTTP. The actual interface to access device attributes and functionality, like temperature values and switching lights, is conceptually decoupled. It can either be served directly through the device or be hosted in the cloud, like in the case of the weather station that regularly pushes its data to a distant server. Beyond REST based interfaces on the device itself or through the cloud, in our approach, the devices can also be accessed over other communication channels available in Web browsers like Web Bluetooth and MQTT. Following the fundamental principles of XML and the Document Object Model in particular, imported elements are put in a tree, i.e. they may reference other components themselves to build up complex user interfaces. Beyond, the user interface elements are shared across instances. Due to Hypertext characteristics, particular elements can be

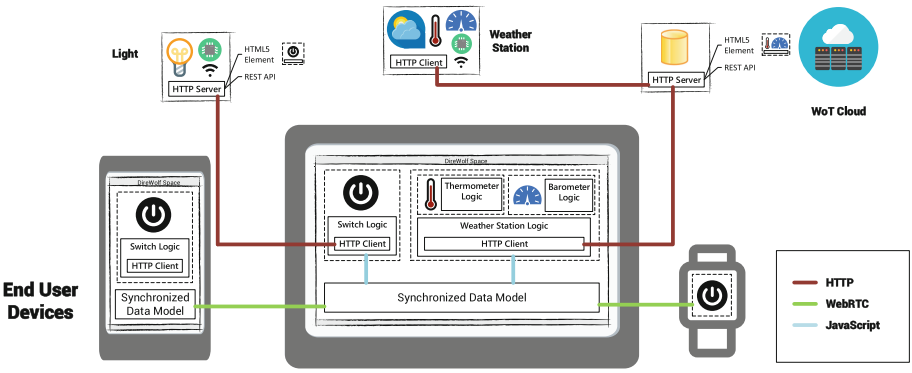

Fig. 1. Overview of the direwolf for the web of things system

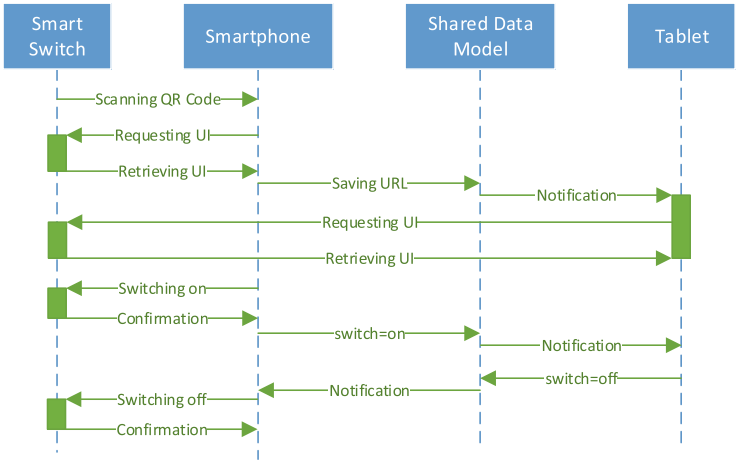

Fig. 2. Sequence diagram of the synchronization

bookmarked and linked either separately or as their combined representation, encapsulated within a structural or layout element.

Once coupled with the DireWolf platform, the user interface elements are aligned in flexible grids according to the concepts of *Responsive Web Design* [18]. All imported elements get access to a shared data model for cross-device synchronization of their state. An optional master flag on the device the element was imported on makes it the primary responsible entity to access the device and saving parameters in the shared model. This way, we avoid redundant requests to constrained devices. Rather, sensor and other values are only accessed once and distributed over the synchronization channel. In the case of the master disconnecting, the functionality is migrated to a new device. A master component could also be responsible for providing cached or interpolated data when certain WoT devices may be temporarily disconnected.

Figure 2 shows a sequence diagram of the overall communication. First, a *Smart Switch* is added to the system that could control an ambient light for instance. Then, the switch is turned on. As can be seen, only one of the DireWolf instances is directly communicating with the physical device, the rest is operating on the shared, synchronized data model. Changes in the data model trigger notifications that in turn may cause requests to the physical device to perform the action. The other way round, this applies to events as well.

4 Implementation of the Platform for the Web of Things

We implemented a prototype using the Web Components group of W3C recommendations that have brought much-needed standardization in the area of componentized frontends for the Web [19]. WoT components are imported using

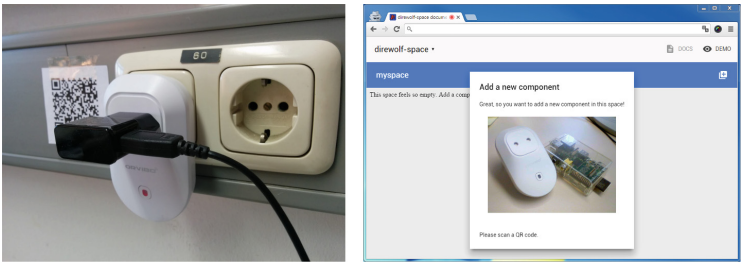

Fig. 3. Screenshot of the scanning process

HTML Imports that define *Custom Elements*. These in turn define their UI elements in *HTML Templates* within their *Shadow DOM*. To provide backwards-compatibility with browsers not yet supporting the standards, a polyfill library is imported, that transparently handles all functionality that is not present on the current platform; if the methods are available natively, they are used instead. Parts of our implementation use the *Polymer*[3] library from Google that adds syntactic sugar on top of the Web Components JavaScript API calls. For example, a complete set of responsive, well-designed user elements are available through Polymer.

DireWolf provides application *spaces* for separating different applications. In our concept, all elements within the same space share a common data model. This is realized with a custom HTML5 element called `direwolf-space` that can be declaratively controlled with attributes. For instance, the `space` attribute defines the space's name. The `direwolf-space` element can be embedded in arbitrary HTML pages; we have already successfully deployed it in WordPress and ROLE SDK instances.

Elements wishing to use the shared data model need to implement the abstract class `DireWolf-Element-Behavior`. A `synced-properties` attribute can then be used to list all parameters of the element that should be synchronized across DireWolf instances. The synchronization layer is implemented using Yjs, our library for synchronizing data structures in a peer-to-peer way [20].

Figure 3 shows a screenshot of the Web application taken on a Laptop. It shows the space called "myspace". A toolbar button on the right opens a dialog that embeds a QR code scanner application. The scanner itself is developed natively with Web technologies. Upon scanning of a code, the interface of the WoT device is added to the space.

5 Conclusion and Future Work

In this article, we have introduced our conceptual extension of the DireWolf framework towards integrating heterogeneous Web of Things devices. The platform is based on state-of-the-art Web Components, thus the application spaces

[3] https://www.polymer-project.org/1.0/.

are embeddable in any kind of (responsive) HTML5 websites. User interface elements can be aligned in tree structures for delivering more complex applications.

To preliminarily validate the conceptual findings and use the results of the implementation in a real-world setting, we have set up a technical evaluation testbed using a variety of mobile devices and a number of commercially available and custom-made smart things connected to the Internet. Our scenario is based on Fig. 1 with a GSM-One WiFi smart socket and a Netatmo weather station. In both cases, designing the UI logic with Web Components could be performed in little time based on the broad documentation of Polymer UI components. The challenge was accounting for the vendor-specific API endpoints in the application logic: While for standards like XMPP and MQTT developers profit of a wide variety of Open Source libraries, both our test appliances required getting familiar with their proprietary REST APIs. Recent initiatives by global players such as Google and Mozilla are currently embedding further communication channels into the Web, including Web Bluetooth and mDNS; these technologies are a valuable add-on for DireWolf.

Because of the late-breaking style of this article, a thorough evaluation of our concept still has to be carried out. We are especially interested in the scalability of our concept, i.e. how many end users can use how many components for which number of WoT devices. Besides, we plan to analyze aspects of usability, accessibility and security. Technical challenges remain in JavaScript dependency management; in the current prototype, all imported elements need to reference the same version of 3rd party libraries to avoid undesired behavior. Module loaders that are currently being standardized in ECMAScript 6, the next generation of JavaScript, may solve this problem in the future. Finally, we envision headless DireWolf clients conceptually acting as microservices for performing the actual connection to WoT devices, while broadcasting results over the shared data model. They could as well serve as gate keepers for verifying access rights of users in collaboration with authentication and authorization providers. We are confident, that our framework can help build future Web infrastructures for a heterogeneous device landscape.

Acknowledgements. The work has received funding from the European Commission's FP7 IP Learning Layers under grant agreement no 318209.

References

1. Daniel, F., Matera, M.: Mashups: Concepts, Models and Architectures. Springer, Heidelberg (2014)
2. Balasubramaniam, S., Lewis, G.A., Simanta, S., Smith, D.B.: Situated software: concepts, motivation, technology, and the future. IEEE Softw. **25**(6), 50–55 (2008)
3. Lieberman, H., Paternò, F., Wulf, V.: End User Development. Human-Computer Interaction Series. Springer, Dordrecht (2006)
4. Burnett, M., Cook, C., Rothermel, G.: End-user software engineering. Commun. ACM **47**(9), 53 (2004)

5. Wulf, V., Jarke, M.: The economics of end-user development. Commun. ACM **47**(9), 41–42 (2004)
6. Sutcliffe, A.: Evaluating the costs and benefits of end-user development. ACM SIGSOFT Softw. Eng. notes **30**(4), 1 (2005)
7. Anderson, C.: The Long Tail: Why the Future of Business Is Selling Less of More. Hyperion, New York (2006)
8. Ogrinz, M.: Mashup Patterns: Designs and Examples for the Modern Enterprise. Addison-Wesley, Upper Saddle River (2009)
9. Guinard, D., Trifa, V.: 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 the 18th International Conference on World Wide Web. ACM, New York (2009)
10. Guinard, D., Trifa, V., Mattern, F., Wilde, E.: From the internet of things to the web of things: resource oriented architecture and best practices. In: Uckelmann, D., Harrison, M., Michahelles, F. (eds.) Architecting the Internet of Things, pp. 97–129. Springer, Heidelberg (2011)
11. de Freitas, A., Nebeling, M., Chen, X.A., Yang, J., Ranithangam, A.S.K.K., Dey, A.K.: Snap-to-it: a user-inspired platform for opportunistic device interactions. In: Proceedings of the 34th Annual ACM Conference on Human Factors in Computing Systems (CHI 2016) (to be published, 2016)
12. Chudnovskyy, O., Nestler, T., Gaedke, M., Daniel, F., Fernández-Villamor, J.I., Chepegin, V., Fornas, J.A., Wilson, S., Kögler, C., Chang, H.: End-user-oriented telco mashups: the OMELETTE approach. In: Proceedings of the 21st International Conference Companion on World Wide Web (WWW 2012 Companion), p. 235 (2012)
13. Govaerts, S., et al.: Towards responsive open learning environments: the ROLE interoperability framework. In: Kloos, C.D., Gillet, D., Crespo García, R.M., Wild, F., Wolpers, M. (eds.) EC-TEL 2011. LNCS, vol. 6964, pp. 125–138. Springer, Heidelberg (2011)
14. Häkkilä, J., Korpipää, P., Ronkainen, S., Tuomela, U.: Interaction and end-user programming with a context-aware mobile application. In: Costabile, M.F., Paternó, F. (eds.) INTERACT 2005. LNCS, vol. 3585, pp. 927–937. Springer, Heidelberg (2005)
15. Cappiello, C., Matera, M., Picozzi, M.: End-user development of mobile mashups. In: Marcus, A. (ed.) DUXU 2013, Part IV. LNCS, vol. 8015, pp. 641–650. Springer, Heidelberg (2013)
16. Chaisatien, P., Prutsachainimmit, K., Tokuda, T.: Mobile mashup generator system for cooperative applications of different mobile devices. In: Auer, S., Díaz, O., Papadopoulos, G.A. (eds.) ICWE 2011. LNCS, vol. 6757, pp. 182–197. Springer, Heidelberg (2011)
17. Kovachev, D., Renzel, D., Nicolaescu, P., Koren, I., Klamma, R.: DireWolf: a framework for widget-based distributed user interfaces. J. Web Eng. **13**(3&4), 203–222 (2014)
18. Marcotte, E.: Responsive Web Design. A Book Apart, New York (2011)
19. Krug, M., Gaedke, M.: SmartComposition: enhanced web components for a better future of web development. In: Proceedings of the 24th International Conference on World Wide Web, pp. 207–210
20. Nicolaescu, P., Jahns, K., Derntl, M., Klamma, R.: Yjs: a framework for near real-time P2P shared editing on arbitrary data types. In: Cimiano, P., Frasincar, F., Houben, G.-J., Schwabe, D. (eds.) ICWE 2015. LNCS, vol. 9114, pp. 675–678. Springer, Heidelberg (2015)