Designing an Application Store for the Internet of Things: Requirements and Challenges[1]

Simon Stastny[1], Babak A. Farshchian[2(x)], Thomas Vilarinho[2]

[1]Norwegian University of Science and Technology, Trondheim, Norway
`stastny.simon@gmail.com`
[2]Stiftelsen SINTEF, Trondheim, Norway
`{babak.farshchian, thomas.vilarinho}@sintef.no`

Abstract. Although things in the Internet of Things contain considerable amounts of software, developers of such software have no standardized means of maintaining, improving and sharing this software as they can do, e.g., with applications on a smart phone. This limitation can hamper user-driven innovation. In this paper we evaluate the usefulness of the "app store" metaphor as a means of sharing and deploying Internet of Things software among makers. We did a set of interviews and a questionnaire-based survey with a sample of makers in various maker communities. We used this data to extract requirements for an application store, using the common "app store" metaphor as a starting point. The app store concept was developed as a proof of concept implementation, and evaluated through feasibility evaluation and focus group evaluation methods. Our findings show that although the app store metaphor is familiar and easy to grasp, there are some fundamental challenges when adapting the metaphor: 1) The difficulty of supporting the diversity in the software and hardware vendor market, 2) The tension between context awareness and the need for pre-configuration and pre-packaging, and 3) usability challenges related to the number of devices and apps.

Keywords: Internet of Things, IoT, app store, application repository, app installation, app deployment, app sharing, ambient intelligence, ubiquitous computing, pervasive computing.

1 Introduction

The Internet of Things (IoT) is defined as "*the pervasive presence around us of a variety of things or objects –such as...tags, sensors, actuators, mobile phones [that] are able to interact with each other and cooperate with their neighbors to reach common goals*"[1]. IoT can be seen as the enabling technology for many of the applications that are envisioned by the Ambient Intelligence (AmI) community. The exponential growth

[1] This is a post-peer-review, pre-copyedit version of an article published in AmI 2015. Lecture Notes in Computer Science, vol 9425. Springer, Cham. The final authenticated version is available online at: https://doi.org/10.1007/978-3-319-26005-1_21

of the IoT can therefore be seen as an opportunity to realize some of these visionary scenarios.

When talking about IoT, the obvious thing that comes to our mind is the *thing* or the physical object. But we should not forget that IoT's value propositions are equally based on the software that runs these things. This software can be present in many forms: embedded, middleware, applications, service composition logic, and management tools [1, 2]. Our interest in this study is related to the embedded software and the management tool used to share, distribute and deploy this software. Using popular denotations from the smartphone domain, we call the collection of these management tools the *IoT app store*. We also call the embedded software that resides in the things for *IoT app*.

IoT apps are in a transition from being "embedded" into IoT things—i.e. being inseparable from and secondary to the thing itself—to becoming an independent and central business asset. In many domains nowadays "firmware upgrades" start getting more attention than product releases—see for instance firmware upgrades for Leica digital cameras. Software engineering practices related to maintaining, upgrading and releasing "firmware" are becoming important topics even in industries such as automotive, where the physical product plays a central role [3]. Some have even gone so far to consider the physical product—the thing—as being secondary to the software that runs on it [4].

When the IoT app becomes a separately tradable asset, it becomes also interesting to look into tools that can support its trade. We already have the popular example of smartphone application stores such as those of Apple, Google and Amazon [5]. These stores have played a central role in grassroots and third-party initiated innovation in the area of smartphone apps [6]. User-driven innovation in IoT apps can be a strong vehicle for a wider uptake of AmI. Preparing a marketplace for sharing such innovation is crucial for nurturing and promoting this innovation [7]. However, due to limitations and complexities that exist in current tools and technologies, such innovation is mostly happening in specialized communities such as among researchers [8, 9] and maker/hackers [10]. As researchers, our interest is drawn to maker/hacker communities, and how their members share their innovations. We believe maker communities, being early adopters [11], can provide us with important insights into how the IoT app landscape will look like in the near future.

In this paper we will present a study of how the app store metaphor might be adapted to IoT. Our research question is "*what are the requirements and challenges for an app store for IoT?*" We have studied a number of makers in order to understand the challenges they face when packaging, sharing and deploying IoT apps. We have used the findings to extract requirements for an IoT app store concept called UbiBazaar. The concept was developed as a paper prototype, evaluated in a focus group, and later implemented as a proof-of-concept prototype. The paper will describe this study by first discussing our method, data and findings from user studies. We will then present the UbiBazaar proof-of-concept implementation, and discuss our findings.

2 Method and Approach

Our study uses the design science research methodology [12], where the research activities are divided into the three cycles of *rigor*, *relevance* and *design* [13]. Here we will shortly describe how we have taken these three aspects of the methodology into account.

To ground our research in existing theory and related work and to clarify our contribution (the rigor cycle in design science) we conducted a systematic literature review. We ran a query in *Scopus* based on common phrases such as IoT, AmI, Pervasive and Ubiquitous computing, software deployment, installation etc. We ended up with 164 hits (October 2014). Ten of these papers were retained after a screening based on title and abstract [inclusion criteria: 1) paper is about deployment in IoT, 2) paper is about app stores in IoT]. In addition, we have studied a number of other papers based on other informal search and snowballing from references in the included papers. Details about the literature study can be found in [14].

In order to increase the relevance of our research (the relevance cycle in design science research) we conducted interviews and a survey with makers. Based on the findings from the literature we conducted four in-depth semi-structured interviews with four experienced makers and developers of IoT projects and software. The interview topics were related to the nature of the IoT projects the interviewees were working on, the methods they used to develop and deploy software, and the challenges they face in the process. Details about the interviews, the interview guide and transcriptions can be found in [14].

Based on the findings from the literature and the interviews we designed and published an online questionnaire. We sent the questionnaire to a number of maker communities (in total 8 communities) and asked their members to participate. In this questionnaire we wanted to investigate the means of software deployment, distribution, and collaborative development. We also wanted to know which IoT platforms are popular among makers. A total of 11 members from these communities responded to the questionnaire, and one of them was further interviewed by us.

Based on these results we designed a paper prototype and a proof of concept implementation of an IoT app store called UbiBazaar (the design cycle in the design science research). The paper prototype was evaluated in a focus group workshop with four experts, the results of which guided the development of the software prototype.

The results from these studies will be presented in the following sections. Section 3 documents the literature study. Section 4 discusses our findings from the interviews and the survey. Section 5 and 6 discuss the results from designing and evaluating UbiBazaar.

3 Related Work

In this section we present an analysis of our findings from the literature. The section is divided into three sub-headings: 1) general discussion of motivation and challenges of using app stores in IoT and similar domains, 2) the challenge of the heterogeneity of platforms and ecosystems, and 3) the challenge of changing context and configuration.

3.1 Software Deployment in IoT

Most non-technical people, when buying an intelligent "thing" or device, never update or change the IoT app that comes with it. This has disadvantages because outdated apps can mean outdated devices. Therefore maintaining, sharing and deploying IoT apps are becoming increasingly common activities. These activities are mostly done in three ways: 1) through proprietary "firmware upgrades", 2) through source code sharing in open communities, and 3) through proprietary app stores.

Firmware upgrades are proprietary and cumbersome methods for updating IoT apps. Most firmware upgrades require a high level of technical knowledge, such as downloading and unpacking files and specialized tools, and transferring files to devices using specialized cables. Many companies are secretive about protocols used to do firmware upgrades and don't allow third parties to develop firmware because of commercial or security concerns. Some industries have been involved in attempts to standardize and open up the firmware upgrade process. Makowski et al. [15] describe a standardized method for upgrading firmware in the Telecommunications Computing Architecture (xTA). Another example is the OSGi platform—used in e.g. automotive industry—that has for many years provided advanced means for deploying so-called bundles onto an OSGi runtime platform [16]. A recent standardization effort is based on using Docker[2], a deployment tool, together with Raspberry Pi. This Docker-based method is demonstrated in our concept of UbiBazaar, to be discussed later.

IoT and its physicality have been a contributing factor to the emergence of strong maker/hacker and Do-It-Youself (DIY) communities in recent years [10]. These communities have gradually developed their own channels of sharing and deploying IoT apps. One of the major channels used is configuration management tools such as Git. Github and specialized portals—such as those of Arduino and Raspberry Pi—are used as channels for distributing open source code for IoT apps. Although popular among makers and DIY communities, this is a complicated method for many people as it involves interacting with code repositories, and configuring and compiling code. As we will see later, the makers we interviewed were looking for better ways of distributing their IoT apps than through code repositories.

The third method of deploying software—i.e. using app stores—is gaining in popularity as app stores in general have become a common household tool for everyone including users [5] and developers [6]. What is probably the main advantage of app stores is their user-friendliness both for providing and consuming packaged apps. App stores are suggested by others as means of deploying IoT apps to general public [7, 17, 18] or for research purposes [9, 19]. There are emerging initiatives for building such app stores (see for instance "The Pi Store" for Raspberry Pi[3] or Android Market for

[2] www.docker.com
[3] store.raspberrypi.com

Wear [4]). These existing initiatives are mainly attempts to repeat the success of smartphone app stores, but do not take into consideration the challenges of IoT, pervasive and ubiquitous computing and AmI such as the heterogeneity of platforms and the context-awareness of IoT apps.

3.2 Heterogeneous Platforms and Ecosystems

The most frequently mentioned challenge in the literature we studied was the heterogeneity of the IoT environments [17, 18, 20, 21]. This heterogeneity is regarded as an issue that is harming the further uptake of IoT innovations, and leading to “a solution that may be outdated quickly” [18]. This heterogeneity makes software deployment harder, as the way a software artifact is deployed or even packaged is in many cases platform-specific. Some works suggest this is the reason why we do not have standardized distribution channels for IoT apps [17]: “The IoT industry doesn’t have a unified hardware and software platform [which] greatly complicates the creation of distribution channels for software applications.”

3.3 Configuration and Context Awareness

A challenge facing IoT app deployment is the need for local configuration of IoT apps due to their varied and changing context. IoT systems are formed by a number of interconnected things, which need to be paired, registered, or configured in order to be able to work together [1]. This configuration is unique for each situation, which makes it difficult to distribute pre-packaged standard IoT apps.

Some researchers [20–23] suggest addressing the challenge of configuration by employing runtime self-configuration mechanisms [24]. Self-configuration can be achieved through context- awareness, i.e. automatically reacting to changes in other parts of the system and the operational environment.

Another approach to configuration and context awareness is to allow users do the configuration through e.g. end user configuration [25, 26]. Both end user configuration and automatic configuration are promising approaches that provide valuable input to how an app store concept can deal with specific context of use for IoT apps. Our approach when developing the concept for UbiBazaar has been to define part of the context—i.e. the device capabilities—handled by the app store, while letting IoT app developers chose the means to configuring the apps once installed.

4 Findings from the Interviews and the Survey

Our informants were involved in IoT projects making systems as diverse as sensor systems for collecting data from crisis situations, ubiquitous smart home systems for monitoring energy consumption, wearable fall detection systems for elderly, and devices augmented with social computing features. Data from 5 in-depth interviews and

[4] www.android.com/wear/

11 answers to survey questionnaires were analyzed using topic-based qualitative data analysis. From this analysis three topics emerged, partly overlapping with our findings from the literature: 1) diversity of deployment platforms, 2) cumbersome distribution and deployment channels, and 3) context awareness vs. pre-configuration.

4.1 Diversity of Deployment Platforms

Most of our respondents use multiple hardware platforms in their projects, including Arduino and Raspberry Pi as the most prolific. The respondents also mentioned other platforms. See **Fig. 2** for an overview of reported platform in our survey.

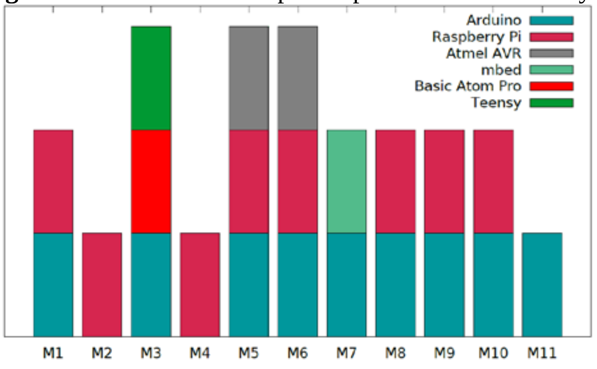


Fig. 2. Overview of the survey informants' usage of hardware platforms.

Another finding is the large number of device variants. A number of these devices—such as Arduino and Raspberry Pi—consist of generic "boards" that can be equipped with sensors, actuators and other peripherals of choice. This means that there is no standard "Arduino device" or "Raspberry Pi device". Each device is potentially unique. Most of the respondents also mentioned that they use multiple platforms within the same project.

The wide variety of hardware platforms results in an even wider variety of deployment tools and mechanisms. While some platforms require use of specialized tools to program a device—e.g. Arduino—other platforms require users to come up with their own ad hoc means of deployment. Some platforms also require physical access to the device—e.g. using a USB cable—while others support remote deployment over wireless network.

4.2 Cumbersome Distribution and Deployment Channels

The difficulty of distributing and deploying IoT apps was mentioned by a number of our informants. In a number of cases the informants were working with projects where the devices were not in their physical vicinity. This meant that they had invented ad hoc means for remote installation of software, which did not seem to be so user-friendly:

- *"I do all deploying myself, in this moment. But I do not have physical access to all of those controllers..."*

- *“It would be nice if this [deployment] could be done wirelessly.”*
- *“...I really could see wireless deployment of upgrades coming down from the cloud, that could be user-transparent, to be useful”*

The informants had some thoughts about scalability of their home-grown deployment methods:

- *“If the prototype gets further developed and maybe commercialized, of course we would need a tool or a procedure to upload a new firmware to the Arduino”*
- *“It would be nice to automate build and deploy updates for the microcontroller...”*
- *“What would be useful is a dedicated server wrapping the build tool that would provide building remotely as a service.”*

Some commented how the IoT apps could be packaged in order to facilitate their sharing and deployment:

- *"If you’re going to produce thousands of Raspberry Pis that are customized for this project then you would produce them all with this image that already has this house-monitoring software installed. The other case is when you have someone who wants to create their own application and deploy it in the Raspberry Pi.”*
- *“Typical deployment takes about six hours [...].If you already have a prepared image for the Raspberry Pi, then the entire deployment takes about one hour”*
- *“[the application on Raspberry Pi] could be Dockerized, The Java component, the GUI, the application that stores the different power which is a web application written in PHP. This all can be packaged as a Docker file, kind of an installation script and then once someone takes a Raspberry Pi and installs Docker and get this Docker file, he will have the service running. However there is still pairing with the smart plugs to be done.”*

Sharing of IoT apps with others is done manually through uploading, downloading and compiling source files. This means that the recipient manually fetches the software—downloads a binary package or source code—and then configures it as wished—typically deploying the software, or doing changes to source code and then building and deploying.

All the informants reported that they use source code repositories to share source code of their software. Some use private code repositories, while others use it as a way of releasing the sources for use by the general public. In addition to using popular services such as Github, respondents frequently mentioned that they share the software on community forums or through their personal website or blogs.

Some respondents mentioned distributing software in form of binaries or built packages. While these are easier for an end user to use, often they cannot be configured. Two of the respondents also mentioned they publish their software on software store such as the Pi Store.

4.3 Context Awareness vs. Pre-configuration

Local configuration emerged as a topic in our findings. In particular, network context for the IoT apps was mentioned:
“The mobile talks to Arduino via Bluetooth and sends commands to it.”

- *“In one deployment, the cottage, there is a Raspberry Pi connected with WiFi USB connector to the WiFi network. In other deployments, Raspberry Pi is connected with a network cable to the router.”*

Other types of configuration mentioned included new sensors or actuators getting connected to a board:

- *“When you deploy for the first time the system needs to be manually configured before you can use it."*
- *“[when a new plug is connected] user needs to assign it in the web interface.”*

A related topic that has to do with local configuration is the awareness that our informants had of the fact that they were part of a DIY prototyping community. They were very clear about the fact that what they did was not commercial product development, that they developed a personal prototype for their personal use, and that –when shared –the product had to be customized by its new users:

- *“If the prototype gets further developed and maybe commercialized, of course we would need a tool or a procedure to upload a new firmware to the Arduino”*
- *“One of the ideas of Arduino is really just prototyping. In essence you use Arduino to easily build your prototype, but afterward if you want to sell a product, you go industrial-wise and you have sort of a personalization process [...] You program the firmware to the board in the factory and you’re done."*
- *“It depends on in which moment you want to do [deployment] - if you want to do this on the prototyping level, or the production level."*

5 UbiBazaar: A Proof of Concept App Store for IoT

In order to further test our findings through design we developed a proof-of-concept implementation of an IoT app store called UbiBazaar. UbiBazaar is based on a focus group evaluation of an early paper prototype. The paper prototype was based on our findings from literature and user studies. We first describe UbiBazaar in this section, before we return to the paper prototype evaluation in the next section.

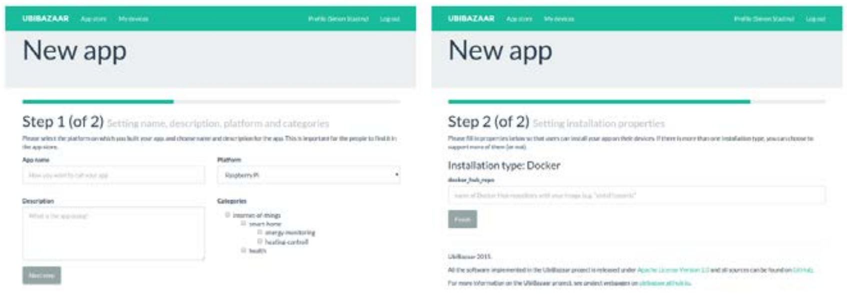
#	Requirement	Rationale
1	UbiBazaar should support the basic functionality and concepts found in popular application markets such as instant wireless installation of apps, and easy-to-use interfaces for developers to share apps.	Since we are testing the usefulness of the "app store" metaphor, it is important that test users can find UbiBazaar concepts similar to those in common app stores, and as easy to use.
2	UbiBazaar should support multiple deployment platforms, and be extensible to other existing or emerging platforms.	Both the literature and our informants point to the fact that IoT consists of multiple hardware and software platforms.
3	UbiBazaar should be aware of and manage the capabilities of the devices belonging to a user.	Our findings show that IoT applications are context-aware. A part of the context that we think can be used by an app store is that of a user's existing devices and their capabilities such as on-board sensors, actuators and network interfaces.

5.1 Start Page and Basic Functionality.

As we are using the app store metaphor as the underlying concept, it is important that the users can get a feeling of visiting an app store when they visit UbiBazaar. The basic functionality for enabling this feeling includes the ability to browse IoT apps and their description, to search for IoT apps based on app category and other criteria, to log in and store a personal profile, including personal apps and devices. In addition, in order to allow support for multiple platforms, the apps are categorized based on the hardware platform they are designed for.

5.2 Sharing New IoT Apps as Developer.

After a user registers, he/she can share IoT apps. The sharing process consists of two steps (see **Fig. 4**): 1) define the basic metadata about the IoT app, and 2) select the hardware platform the app can run on.


UBIBAZAAR App store My devices Profile (Simon Stastny) Log out

New app

Step 1 (of 2) Setting name, description, platform and categories

Please select the platform on which you built your app and choose name and description for the app. This is important for the people to find it in the app store.

App name: How you want to call your app

Platform: Raspberry Pi

Description: What is the app doing?

Categories:
- internet of things
  - smart home
    - energy monitoring
    - heating control
  - health

Next step

UBIBAZAAR App store My devices Profile (Simon Stastny) Log out

New app

Step 2 (of 2) Setting installation properties

Please fill in properties below so that users can install your app on their devices. If there is more than one installation type, you can choose to support more of them (or not).

Installation type: Docker

docker_hub_repo: name of Docker Hub repository with your image (e.g. [illegible])

Finish

UbiBazaar 2015.

All the software implemented in the UbiBazaar project is released under Apache License Version 2.0 and all sources can be found on GitHub.

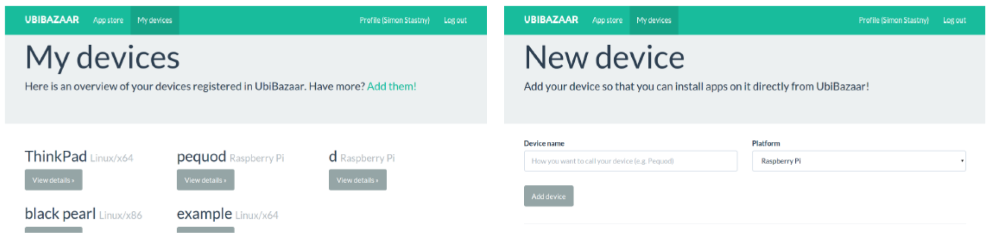
For more information on the UbiBazaar project, see project webpages on ubibazaar.github.io.

(a) Step 1. (b) Step 2.

Fig. 4. Adding a new IoT app consist of two steps: 1) provide common metadata, and 2) select hardware platform.

5.3 **Maintaining a set of Device Definitions**.

UbiBazaar allows the user to maintain a set of devices (see **Fig. 5**). The devices are given a name and a description and are assigned a platform type, e.g. Arduino or Raspberry Pi. Upon the registration of a device, a pairing and authentication process is performed. It consists of installing an *installation manager* on the device (see later section on the UbiBazaar architecture), and pairing this installation manager with UbiBazaar server. After this pairing is done, the device can communicate with UbiBazaar in order to query for content and to install/uninstall apps.


UBIBAZAAR App store My devices Profile (Simon Stastny) Log out

My devices

Here is an overview of your devices registered in UbiBazaar. Have more? Add them!

ThinkPad Linux/x64 View details ›

pequod Raspberry Pi View details ›

d Raspberry Pi View details ›

black pearl Linux/x86

example Linux/x64

UBIBAZAAR App store My devices Profile (Simon Stastny) Log out

New device

Add your device so that you can install apps on it directly from UbiBazaar!

Device name: How you want to call your device (e.g. Pequod)

Platform: Raspberry Pi

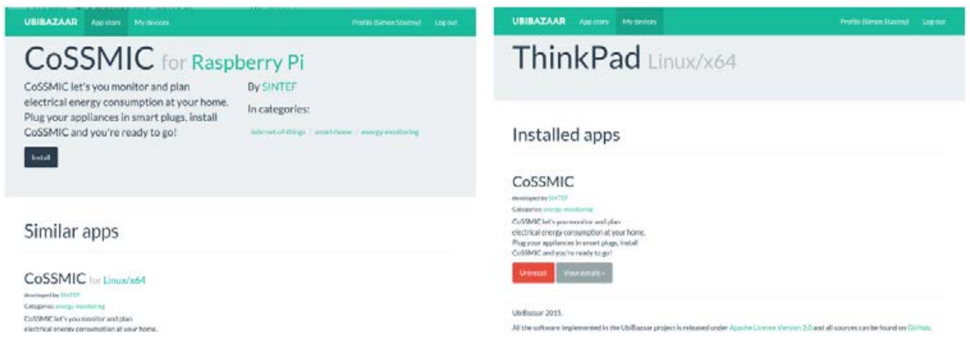
Add device

(a) Device overview. (b) Adding a new device.

Fig. 5. Defining and maintaining a set of devices.

5.4 **Installing an App on a Device.**

UbiBazaar allows the user to browse for IoT apps that are compatible with a specific platform—e.g. Raspberry Pi—and chose a device of that type to install the app (see **Fig. 6**). UbiBazaar also allows the user to maintain the apps that are installed on each device. Some devices, such as Raspberry Pi, allow multiple IoT apps on-board while others—e.g. Arduino—do not.



CoSSMIC for Raspberry Pi

CoSSMIC let's you monitor and plan electrical energy consumption at your home. Plug your appliances in smart plugs. Install CoSSMIC and you're ready to go!

By SINTEF

In categories:

internet-of-things / smart-home / energy-monitoring

Similar apps

CoSSMIC for Linux/x64

developed by SINTEF

Categories: energy-monitoring

CoSSMIC let's you monitor and plan electrical energy consumption at your home.

ThinkPad Linux/x64

Installed apps

CoSSMIC

developed by SINTEF

Categories: energy-monitoring

CoSSMIC let's you monitor and plan electrical energy consumption at your home. Plug your appliances in smart plugs. Install CoSSMIC and you're ready to go!

UbiBazaar 2015.

All the software implemented in the UbiBazaar project is released under Apache License Version 2.0 and all sources can be found on GitHub.

(a) App detail. (b) Device detail.

Fig. 6. Installing an IoT app on a device.

5.5 UbiBazaar Architecture.

Fig. 7 shows UbiBazaar's architecture. It consists of a central *UbiBazaar server* hosting a number of services for maintaining users and their personal profiles such as apps and devices (upper part of the figure). The server also includes the *web front-end* that was described in the previous sections. In addition to the server, a number of *installation managers* are being developed. Installation managers are responsible for implementing the communication between the devices—the things—and UbiBazaar. This communication is necessary for querying the devices about their capabilities and about the IoT apps they host. **Fig. 7** shows how the installation managers for Raspberry Pi and Arduino are implemented. Since Raspberry Pi is a full-fledged Linux-based device with network capabilities, the device (thing) itself can host the installation manager. Arduino is a much simpler device and for this reason we have implemented the installation manager for Arduino as an Android app [27]. This app then communicates with Arduino devices, using Bluetooth to send IoT apps to these devices [28]. We have emphasized the definition and use of open APIs for communication between the components. These APIs are defined and documented in the project web site [29].

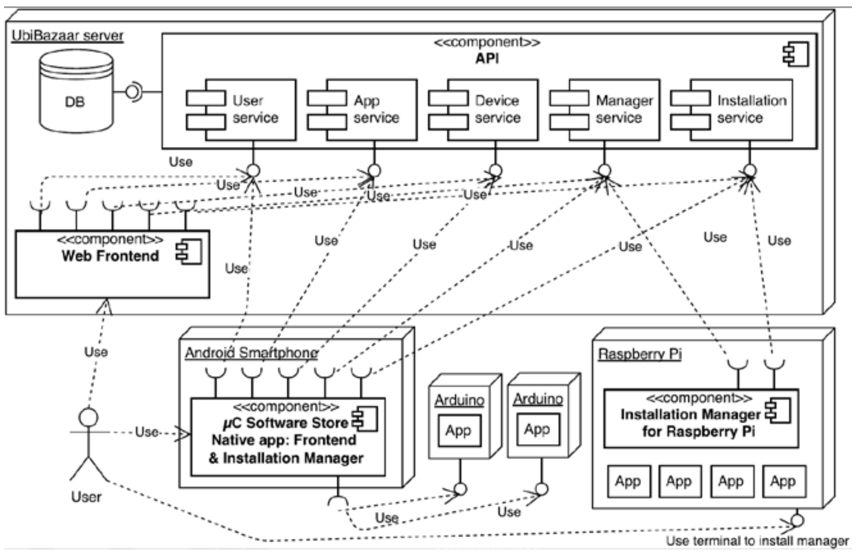


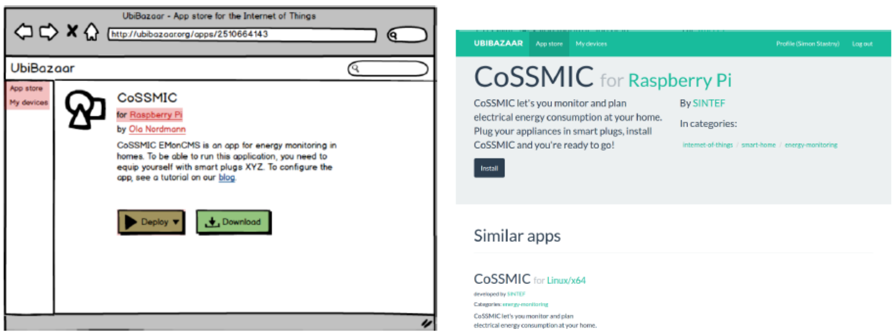
Fig. 7. UbiBazaar architecture.

6 Evaluation

Based on findings from our field and literature studies, the initial concept of UbiBazaar was developed as a paper prototype using the Balsamiq tool[5]. This prototype was evaluated by a focus group of four makers (different than those who were initially interviewed). The results from this focus group were used to refine the concept before it was implemented as a reference implementation that was described in the previous section. This section will provide an overview of the findings from this evaluation of the paper prototype. The main concepts of IoT apps and devices were already demonstrated in this paper prototype. In addition, we also demonstrated the installation and pairing with devices of the type Raspberry Pi.

Streamlining of Concepts and Terminology- One of the main findings from the evaluation was the need to simplify the concepts used in the prototype. For instance, we eliminated the use of the concept "Deploy" and "Download" and used instead concepts such as "Install". See **Fig. 8**.

[5] http://balsamiq.com/


UbiBazaar - App store for the Internet of Things

http://ubibazaar.org/apps/2510664143

UbiBazaar

App store

My devices

CoSSMIC

for Raspberry Pi

by Ola Nordmann

CoSSMIC EMonCMS is an app for energy monitoring in homes. To be able to run this application, you need to equip yourself with smart plugs XYZ. To configure the app, see a tutorial on our blog.

Deploy

Download

UBIBAZAAR

App store

My devices

Profile (Simon Stastny)

Log out

CoSSMIC for Raspberry Pi

CoSSMIC let's you monitor and plan electrical energy consumption at your home. Plug your appliances in smart plugs, install CoSSMIC and you're ready to go!

By SINTEF

In categories:

internet-of-things / smart-home / energy-monitoring

Install

Similar apps

CoSSMIC for Linux/x64

developed by SINTEF

Categories: energy-monitoring

CoSSMIC let's you monitor and plan electrical energy consumption at your home.

(a) Paper prototype. (b) Software prototype.

Fig. 8. Moving from "Deploy" and "Download" to "Install".

Device Pairing and Authentication- The way UbiBazaar handles the first time pairing with devices was discussed by the focus group. The discussion was due to the difficulty of the concept, but also due to practical issues such as firewall configurations. The original mechanism of device pairing and authentication was based on downloading a generic installer, installing the software and then configuring it manually. The focus group found this too complicated for the user and suggested to simplify the process. For the final solution, installation script with built-in credentials and auto-configuration were used. For Arduino, Standard Bluetooth pairing is done on the Android device [28].

Managing Devices- The participants discussed how it will look like when the user has many devices that can communicate with each other. Currently UbiBazaar maintains a flat list of devices. Specifying device capability is not supported. The participants wished to see some way of allowing the user tell UbiBazaar what devices can communicate with each other and how.

Trust and Security Concerns- Trustworthiness in IoT app stores has been addressed by previous research [30] and was also a big concern for the focus group participants. IoT has a potential to influence our daily lives in ways we may not even be able to imagine yet. It is necessary to ensure that IoT apps, many of which will handle our personal data and control our lives to a certain extent, are to be trusted and prevent misuse by third parties. Although simple authentication is implemented in our proof of concept, more research needs to be done in this area.

Social and Collaborative Aspects- The participants meant that UbiBazaar should not only be an app catalogue but also a social platform with features such as ratings and comments. Since IoT apps need local customizations much more than current smartphone apps, the participants envisaged a more central role for social and collaborative aspects in order to support collaborative customization.

7 Discussion

Using the app store metaphor in the context of IoT is an attractive idea, as witnessed by both commercial and research-based initiatives discussed in this paper. The need for an app store-like tool was raised in our user studies. UbiBazaar concept was easily understood and appreciated by our focus group participants. The metaphor is familiar, and has shown to be successful in creating and maintaining sustainable ecosystems. It is therefore natural to think about reusing the concept in the IoT domain. In particular, some properties of IoT—e.g. large numbers of devices, devices without a management interface, the need for remote management and so on [31]—would mean that a management tool such as an app store is the answer. However, our study shows that there are some fundamental challenges facing this hypothesis.

The first challenge is that of the heterogeneity of hardware and software platforms. Compared to the heterogeneity that we can see in IoT, even the smartphone platform market seems very homogeneous. After more than a decade of smartphone development we have not yet arrived at a common platform. So the journey towards a common platform might be much longer for IoT. There are strong commercial interests connected to app stores and the ecosystem surrounding them [32] that point in the direction of even stronger proprietary ecosystems in the future. This will be a challenge for an IoT app store that aims to support a heterogeneous set of things running different software platforms. UbiBazaar architecture is a step in bridging the gap, but will have limitations in supporting commercial, proprietary platforms.

The second challenge is that of context awareness. Smartphone context can arguably be said to be much simpler than that of a typical IoT system consisting of tens of devices each running various IoT apps. Uncritical application of app store concept to such scenarios will mean users will get overloaded with configuration and re-configuration tasks. We need to build on findings from both context-aware and autonomous computing [24], but also use research done in the area of end user programming and end user configuration [26]. The answer will probably lie in combining what the app store can support, what the user can configure, and what the IoT system can/should automate. The goal should be to work on a usable and useful tool for the user. Our approach in UbiBazaar has been to keep UbiBazaar in charge of maintaining device capabilities, and allow app developers decide the type of context-awareness and end user configuration at the app level.

Our study has resulted in findings related to usability. What does it mean to deploy, install or update an IoT app? How do we update remote devices? How do we know which device we are updating? See [33] for a wide range of such issues that can apply to IoT app stores. More research is needed within IoT app store usability.

We believe that maker communities form an ideal setting to do further research on these issues. We see a need for an IoT app store among the makers we have talked to. We also see makers as early adopters of technologies that will be available in every household in the not so far future.

8 Conclusions

In this paper we have studied IoT apps and IoT app stores. We have presented empirical evidence from user studies and the literature about the feasibility of using the app store metaphor in the IoT domain. We have presented a proof-of-concept implementation of an IoT app store called UbiBazaar. UbiBazaar is implemented as an open source project [29] and will be used as platform for further research in this area.

Our surveys (n=11) and interviews (n=5) should be repeated in order to refine and generalize the findings. We believe however that the nature of the practices we have studies is such that that most makers will have the same challenges as those we have studied. Our future research will include studies of how UbiBazaar will be used by makers.

We have focused our empirical investigations towards maker communities. Our assumption has been that the needs of makers, being early adapters, can be generalized to those of non-makers. We believe this is a hypothesis that can be tested in future research, by evaluating UbiBazaar and related concepts with non-makers.

Our reference implementation UbiBazaar is only an early proof-of-concept prototype and needs to be extended in a number of directions. Of most interest for us are the addition of support for collaboration, and the addition of functionality for specifying device capabilities in a more detailed manner as a means for specifying a context for IoT apps. We also think it will be important to add support for a wider variety of platforms, in particular full support for Arduino.

Acknowledgements. This research was funded by the European FP7 projects OPTET (317631) and CoSSMic (608806). We thank AmI 2015 reviewers for constructive comments to the draft of the paper.

9 References

1. Atzori L, Iera A, Morabito G (2010) The Internet of Things: A survey. Comput Networks 54:2787–2805.
2. Ebert C, Jones C (2009) Embedded software: Facts, Figures and Future. IEEE Comput 4:42–52.
3. Broy M (2006) Challenges in automotive software engineering. Proc 28th Int Conf Softw Eng 33–42.
4. Shih WC (2015) Does Hardware Even Matter Anymore? Harv. Bus. Rev.
5. Cuadrado F, Dueñas J (2012) Mobile application stores: Success factors, existing approaches, and future developments. IEEE Commun Mag 50:160–167.
6. Holzer A, Ondrus J (2011) Mobile application market: A developer’s perspective. Telemat Informatics 28:22–31.
7. Kortuem G, Kawsar F (2010) Market-based user innovation in the Internet of Things. 2010 Internet Things 1–8.
8. Gellersen H, Kortuem G, Schmidt A, Beigl M (2004) Physical prototyping with smart-its. IEEE Pervasive Comput 3:74–82.

9. Cramer H, Rost M, Belloni N, et al. (2010) Research in the large. using app stores, markets, and other wide distribution channels in Ubicomp research. In: Proc. 12th ACM Int. Conf. Adjun. Pap. Ubiquitous Comput. - Ubicomp ’10. ACM Press, New York, New York, USA, p 511
10. Lindtner S, Hertz GD, Dourish P (2014) Emerging Sites of HCI Innovation: Hackerspaces, Hardware Startups and Incubators. Proc SIGCHI Conf Hum Factors Comput Syst 439–448.
11. Rogers EM (2010) Diffusion of innovations. Simon and Schuster.
12. Hevner AR, Salvatore T. March, Park J, Ram S (2004) Design Science in Information Systems Research. MIS Q 28:75–105.
13. Hevner AR (2007) A Three Cycle View of Design Science Research. Scand J Inf Syst 19:87–92.
14. Stastny S (2014) UbiBazaar - App store for the Internet of Things. Student thesis. Norwegian University of Science and Technology. Trondheim, Norway
15. Makowski D, Jab G, Perek P, et al. (2013) Firmware Upgrade in xTCA Systems. IEEE Trans Nucl Sci 60:3639–3646.
16. Parrend P, Frenot S (2007) Supporting the secure deployment of OSGi bundles. 2007 IEEE Int Symp a World Wireless, Mob Multimed Networks, 33:1–6.
17. Munjin D, Morin J-H (2012) Toward Internet of Things Application Markets. 2012 IEEE Int Conf Green Comput Commun 156–162.
18. Svendsen RM, Castejon HN, Berg E, Zoric J (2011) Towards an integrated solution to Internet of Things - a technical and economical proposal. In: 2011 15th Int. Conf. Intell. Next Gener. Networks. IEEE, pp 46–51
19. Davies N (2011) Beyond Prototypes, Again. IEEE Pervasive Comput 10:2–3.
20. Medvidovic N, Malek S (2007) Software deployment architecture and quality-of-service in pervasive environments. In: Int. Work. Eng. Softw. Serv. ACM Press, New York, New York, USA, pp 47–51
21. Hoareau D, Mahéo Y (2006) Middleware support for the deployment of ubiquitous software components. Pers Ubiquitous Comput 12:167–178.
22. Zheng D, Wang J, Han W, et al. (2006) Towards A Context-Aware Middleware for Deploying Component-Based Applications in Pervasive Computing. In: 2006 Fifth Int. Conf. Grid Coop. Comput. IEEE, pp 454–457
23. Sung BY, Kumar M, Shirazi B (2005) Flexible and Adaptive Services in Pervasive Computing. Adv Comput 63:165–206.
24. Kephart JO, Chess DM (2003) The vision of autonomic computing. Computer 36-1:41.
25. Danado J, Paternò F (2014) Puzzle: A mobile application development environment using a jigsaw metaphor. J Vis Lang Comput 25:297–315.
26. Paternò F, Tetteroo D, Markopoulos P, et al. (2015) End-User Development in the Internet of Things Era. CHI’15 Ext Abstr.
27. Eriksen J et al. (2013) mC Software Store. Student report. Norwegian University of Science and Technology. Trondheim, Norway
28. Eie AB et al. (2012) oSNAP- Open Social Network Arduino Platform. Student report. Norwegian University of Science and Technology. Trondheim, Norway
29. Stastny S (2015) UbiBazaar project at Github.com. http://ubibazaar.github.io/.

30. Kang K, Pang Z, Xu L Da, et al. (2014) An Interactive Trust Model for Application Market of the Internet of Things. IEEE Trans Ind Informatics 10:1516–1526.
31. Andersson J (2000) A deployment system for pervasive computing. In: Proc. Int. Conf. Softw. Maint. ICSM-94. IEEE Comput. Soc. Press, pp 262–270
32. Eaton B, Elaluf-Calderwood et al. (2015) Distributed Tuning of Boundary Resources: The Case of Apple's iOS Service System. MISQ 39:217–243.
33. Bellotti V, Back M, Edwards WK, et al. (2002) Making sense of sensing systems: five questions for designers and researchers. In: Proc. SIGCHI Conf. Hum. factors Comput. Syst. Chang. our world Chang. ourselves CHI 02. pp 415–422