A Middleware for Implicit Human-Computer Interaction Across IoT Platforms

Ekene Attoh
WISE Lab, Vrije Universiteit Brussel
Brussels, Belgium
eattoh@vub.be

Beat Signer
WISE Lab, Vrije Universiteit Brussel
Brussels, Belgium
bsigner@vub.be

ABSTRACT

The Internet of Things is increasingly becoming a part of our daily life. With popular task automation or IoT platforms such as *If This Then That (IFTTT)*, users are able to define rules to enable interactions between smart devices in their environment and thereby improve their daily lives. However, rules authored by users on these popular task automation or IoT platforms are often tied to the platform or even to the specific devices for which they have been configured, implying a vendor lock-in for users. Therefore, switching to a different task automation or IoT platform or the introduction of new devices might demand for a re-creation of the rules that are specific to the new platform and devices. In order to address this problem, we need human-computer interaction that works across IoT platforms, in particular with the proliferation of IoT services in domains such as smart health where a user's quality of life might depend on an uninterrupted service offered by a platform. In this paper, we propose an architecture to enable implicit human-computer interaction across IoT platforms by introducing the necessary concepts providing users ownership and control over their IoT data and rules.

CCS CONCEPTS

• **Human-centered computing → Ubiquitous and mobile computing systems and tools**.

KEYWORDS

Internet of Things, cross-platform IoT, interoperability, Solid

ACM Reference Format:
Ekene Attoh and Beat Signer. 2021. A Middleware for Implicit Human-Computer Interaction Across IoT Platforms. In *Adjunct Proceedings of the 2021 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2021 ACM International Symposium on Wearable Computers (UbiComp-ISWC '21 Adjunct), September 21–26, 2021, Virtual, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3460418.3479311

1 INTRODUCTION AND RELATED WORK

The avoidance of vertical IoT silos and the enabling of cross-platform interoperability has been investigated in a number of research projects. Li et al. [4] identified the problem of vendor lock-in in *end-user development*, whereby manufacturers provide proprietary software to enable users to create IoT rules for their smart devices. However, the rules created via this software can only be executed on devices of the manufacturer's ecosystem. The popular IFTTT[1] platform enables users to create rules for devices of various manufacturers, but only certain devices and services by partners and a subset of the commonly used functionality are supported [4].

The solution proposed by Li et al. enables users to author rules for their IoT devices by demonstrating interactions between smart devices using their mobile phone. This is achieved via an Android application which enables users to create automation scripts which are composed by recording the actions they perform on the mobile application of their smart devices. The scripts can then be triggered to perform the actions that were demonstrated by the user using a source from another application such as a notification from a motion sensor application or even the launch of another application. The client component of the proposed system is an Android application which has been designed to work on any smartphone with Android 4.4 or higher. We infer that these scripts are stored in memory on the mobile device itself, introducing a user lock-in as the created rules will only be available on the mobile device. Further, recorded actions will just be valid for the devices in the environment where the configuration took place and the scripts will not work on other platforms such as iOS or Windows.

Some EU-led initiatives have also sought to tackle the interoperability issue of IoT platforms. One such project, called *symbIoTe*, aims to address the problem by creating a framework which enables the discovery and sharing of resources for rapid cross-platform application development and facilitates the blending of the next generation of smart objects with surrounding environments. symbIoTe aims to provide a unified view on various platforms and their resources, so that application designers and developers have a transparent view of these resources and can easily access them [7]. Another project called *bIoTope* proposes a marketplace enabling developers and businesses to discover IoT services. Platforms can expose their services as Open Messaging Interface (O-MI)[2] nodes, making them discoverable via the marketplace. Developers can then gain access to these services by paying for them via the marketplace and directly connecting to the O-MI service nodes [3]. As can be seen, these projects do not focus on the interoperability of user-defined rules across platforms, but rather on abstracting the services provided by different platforms and making them accessible to developers when creating their solutions.

[1]https://ifttt.com
[2]http://www.opengroup.org/iot/omi/p8.htm

Corno et al. [1] introduced *EUPont*, a high-level semantic model for end-user IoT development. With EUPont, users are able to create rules at a higher level of abstraction, eliminating the need to configure them for specific devices or environments. For example, an IFTTT rule such as *"If my smart sensor X detects that I am home and the outside temperature is less than 10 degrees, then turn on my smart heater H"*, will have to be configured multiple times for each new environment and device. With EUPont, this rule can be transformed to *"If I am in an indoor place and the outside temperature is less than 10 degrees, then start heating the indoor place"*. With this higher level of abstraction, any device capable of heating will simply be used to perform the required action. We see this solution as a good starting point for our work on cross-platform interactions, but also note some limitations. Most IoT users already have tools which they are familiar with and have existing rules they have authored using these tools. We posit that users should have the flexibility to use any tool of their choice and they should not have to switch to a completely new solution in order to author their IoT rules. We also note that the functionality exposed by task automation or IoT platforms, such as the one proposed by Corno et al., are often quite simple and prevent users from creating richer reusable situations [8]. For example, in state-of-the-art platforms the *If* side of a rule such as *"If my heart rate is higher than 100 bpm, then change the colour of the smart light in my living room to red"* will have to be re-defined each time the user wants to perform an action based on their heart rate value. With the solution proposed by Trullemans et al. [8], the *If* side can also lead to the definition of a reusable *situation* rather than simply triggering an action. For example, the condition *"If my heart rate is higher than 100 bpm"* might lead to the situation *"My heart rate is high"*. This situation can then be used on the *If* side to create other rules, without the need to re-define the threshold for a high heart rate. This functionality, which is missing in other state-of-the-art platforms, can serve as a building block for enabling cross-platform interactions. A user having defined what a high heart rate is on *platform A* can simply rely on this definition and use the high-level abstraction (*"If my heart rate is high"*) on a different *platform B*.

Corno et al. also stated that 295'156 IFTTT rules have been automatically translated to EUPont rules. This means that rules which were authored using another platform cannot yet be used on the proposed system. Another limitation we wish to point out is the fact that the rules are stored on the EUPont server. It is not mentioned whether there is any possibility to transfer the rules to a different system, thus the EUPont server retains ownership of the created rules, as it is the case with existing popular IoT platforms.

A final limitation is the fact that with this higher level of abstraction, users will lose the ability to define specific environments and/or devices where their rules should be executed. Using this model, rules will be executed anywhere a device supporting the required action is detected. This could lead to unpleasant scenarios, such as personal rules being executed on public devices and a notification to take some medication being displayed on a public screen.

2 PROBLEM STATEMENT

Based on the analysis of existing work and the state-of-the art in implicit human-computer interaction for IoT environments, we identified the following two major problems to be addressed in our research in order to enable interactions across IoT platforms.

2.1 Rule Ownership

As stated in Section 1, state-of-the-art task automation and IoT platforms retain the ownership of the user-generated rules. This is due to the fact that data is coupled to the platform and thus stored on the platform. The authors in [1, 4] proposed systems which enable users to create rules in a novel way, but both systems retain ownership of the rules created due to the fact that the rules are tied to the platform. We note that with the solution in [4], the created rules will be stored in the mobile device application, while the authors in [1] make no mention on whether the rules created by their system can be completely transferred to a different system. This platform data ownership might thus prevent users from running their rules across different platforms, since they are unable to move their rules from the platform they were created on.

2.2 Rule Interoperability

The work in [3, 7] aims to tackle platform interoperability by making IoT services discoverable and accessible to developers and businesses through a marketplace. However, this still leaves the interoperability of (end-)user rules an open issue. The solutions proposed by the authors do not address the fact that a rule created by a user on one platform and for specific devices might be completely unusable on a different platform and for a different set of devices. Since platforms might have differing means for enabling users to compose rules for their IoT devices—*platform A* might for instance use a *trigger-action* programming paradigm to allow users declare rules [9] while *platform B* uses a *programming by demonstration* paradigm [4]—rules which have been created on *platform A* might not be understood by *platform B* and vice versa. Therefore, tackling this issue is vital to improve the user experience in IoT interactions. Corno et al. [1] introduced a high-level semantic model for IoT rules but as we demonstrated in Section 1, it lacks support for richer rule definitions such as the concept of *situations* introduced by Trullemans et al. [8], which might improve cross-platform interoperability.

3 METHODOLOGY

Our research is conducted by following the Design Science Research Methodology (DSRM) for information systems research by Peffers et al. [5]. This methodology fits well with our research as it was designed for the creation of theoretical artefacts such as models and algorithms as well as more concrete outcomes including software frameworks and user interfaces. As our research covers end-user software, (end-)user interfaces as well as information storage, management and visualisation, the DSRM's focus on information systems research makes it a suitable methodology. The DSRM process model builds on prior research in design science and consists of six steps, including the *problem identification and motivation*, the *definition of objectives for a solution*, the *design and development*, the *demonstrating*, the *evaluation*, and the *communication*.

As described earlier, we have already identified the main problems of *rule ownership* and *rule interoperability*, and motivated our planned research in these areas. Our objective is to tackle these

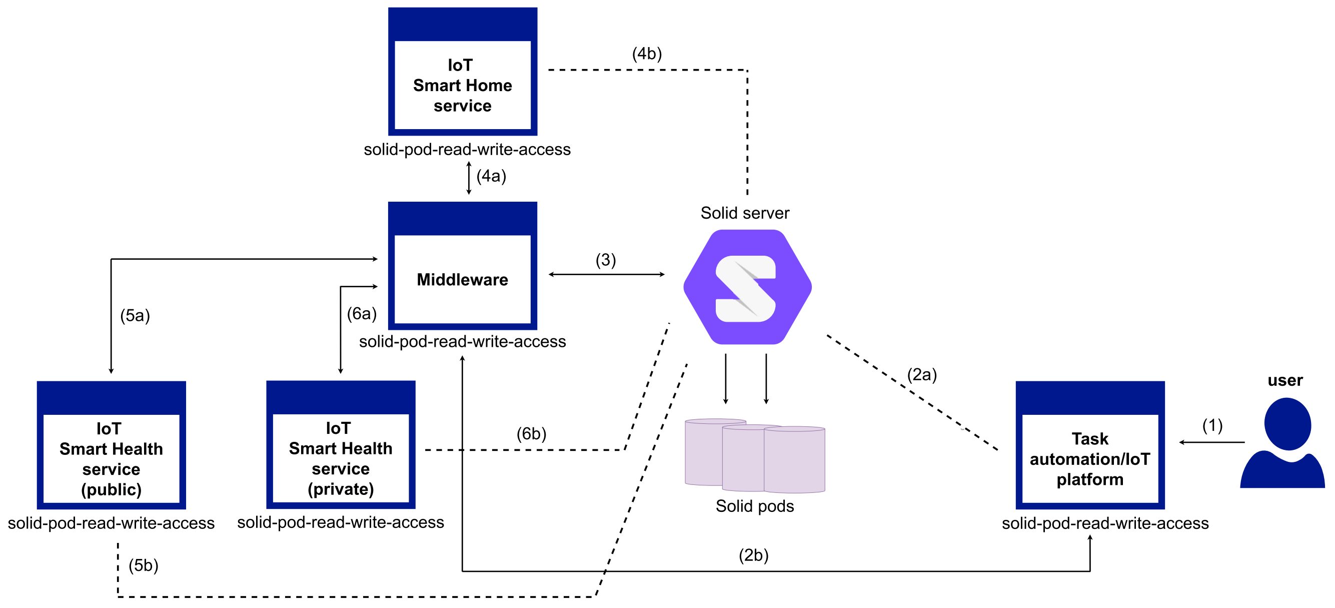
IoT Smart Home service
solid-pod-read-write-access
(4a)
(4b)
Middleware
solid-pod-read-write-access
(3)
Solid server
(5a)
(6a)
(6b)
(2a)
(1)
user
IoT Smart Health service (public)
solid-pod-read-write-access
IoT Smart Health service (private)
solid-pod-read-write-access
Solid pods
Task automation/IoT platform
solid-pod-read-write-access
(5b)
(2b)

Figure 1: Cross-Platform IoT architecture

problem areas by proposing a solution that enables users to maintain the ownership of their rules and offers them the flexibility to either retain their current platforms for rule authoring or use any platform of their choice. Our proposed solution will further enable users to have their rules executed across different platforms of their choice to support cross-platform IoT solutions. Our design and development step consists of creating the necessary architectural design in order to conceptualise how such cross-platform IoT systems might be realised. Based on the proposed architecture, we then plan to develop a proof-of-concept middleware for implicit human-computer interaction across IoT platforms. The utility and value of our conceptual architecture will be demonstrated through concrete use cases and the usefulness of the proof-of-concept middleware solution will be demonstrated in a similar way. We further plan to evaluate our conceptual architecture and proof-of-concept middleware implementation by conducting a user study in order to gain feedback on its usefulness and get further insights about its limitations and potential future improvements. Last but not least, we plan to publish our findings and contributions at international peer-reviewed conferences and journals.

4 PROPOSED SOLUTION

Our proposed architecture, which we also use to introduce the core components that we propose to tackle the problems identified in Section 2, is illustrated in Figure 1. We explain the components with the help of a hypothetical user scenario of a user making use of a smart health IoT service. They expect that they will be reminded to take their medication when their heart rate is above 100 bpm and have the following requirements for a solution:

- The user wants to have exclusive access to their rules containing sensitive information about their health
- The user wants to be reminded by a light bulb flashing red in their living room
- The user is expected to go on holiday but still expects the same kind of reminder (light bulb flashing red) when it is necessary to take their medication at the holiday destination

4.1 Rule Storage

In the context of the World Wide Web, researchers have identified a similar vendor lock-in problem as we identified in Section 2. With so-called *centralised web platforms*, users cannot easily move data between platforms or switch between similar applications that could reuse their data. The *Solid* project [6], which stands for *Social Linked Data*, aims to provide data independence as well as simple yet powerful data management mechanisms. With Solid, users store their data in an online storage space called *personal online data store (pod)*. A pod is a web-accessible storage service which can be deployed on servers hosted by users themselves or by pod providers [6]. A user is able to have multiple pods from one or different providers and they are the exclusive owner and administrator of their pods. Applications no longer store data themselves but can get the data they require from Solid pods as shown in Figure 2. The user is in control of their pod(s) and can grant or revoke applications access to their pods.

We thus use the concept of pods as offered by Solid to separate user-defined IoT rules and even IoT data from task automation and IoT platforms. This separation is a first step in addressing the vendor lock-in of state-of-the-art IoT solutions, as a user's Solid pod becomes the exclusive data store. As shown in Figure 1, the user creates a rule *"If my heart rate is above 100 bpm, make the smart light bulb in my living room flash red"* using their task automation platform (1). There are then two possible flows following the rule creation. With the first flow (2a), the user's task automation platform enables them to define rules with a high level of abstraction such as with EUPont [1]. Note that the user has granted the task automation platform write permission to their Solid pod such that

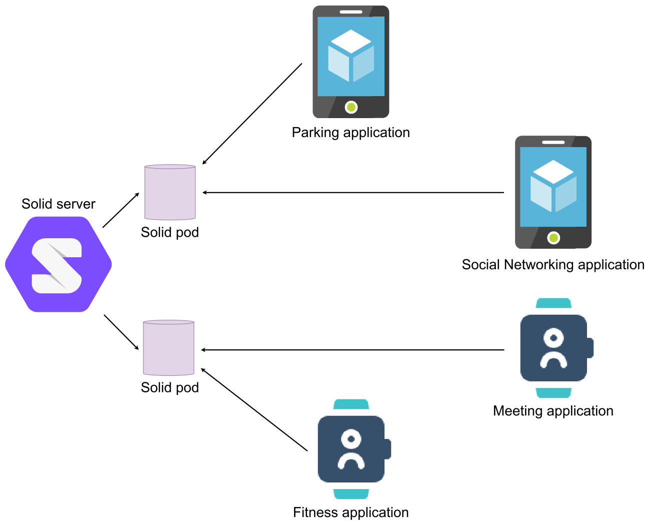

Figure 2: Applications making use of Solid pods

it can directly store rules in the pod. In the second flow (2b), the user's task automation platform enables them to define rules in a proprietary format. In order to enable these proprietary rules to work across different platforms, they have to pass through our middleware component which is explained in the next section.

4.2 Middleware

A next step for tackling the problem area of *rule interoperability* is the introduction of a middleware component. In order to enable rules created by the user to work on different platforms, we propose the flow where the user's task automation system sends the proprietary rule to the middleware component (2b). In order to meet the user's requirement of having exclusive ownership and administrative rights of their rules, we propose that the middleware component then translates their rule to a high-level format such as EUPont [1]. We plan to introduce the missing concepts we identified in this high-level format, including the situations proposed in [8] as well as regaining the possibility of specifying environments and/or devices where certain user-defined rules should be executed.

It is assumed that the middleware component has been granted write access to the user's Solid pod such that it is able to store the translated rules (3). In our example, the translated stored rule might be *"If my heart rate is high, flash a red light in my current location"*. The user can now grant read-write access to the rule in their pod to their smart home and smart health IoT applications. There are two possible flows for each IoT application. For the flows (4b) and (5b), it is assumed that the IoT applications understand the high-level rule format by default and are thus able to directly read and execute the rules from the pod. This means that the smart health service (public) will read the rule *"If my heart rate is high, flash a red light in my current location"* from the user's pod. In order to trigger a flashing red light, we need a way to notify the smart home service. We will investigate different approaches on how this could be done but one proposal is, given that the smart home and smart health services understand the high-level format, the smart health service can simply trigger the lighting action in the smart home service. For the flows (4a) and (5a), it is assumed that the IoT applications do not understand the high-level rule format by default. We thus propose that the middleware component translates the high-level rules back to the proprietary format understood by the IoT applications. This means that the smart health service (public) having read the rule *"If my heart rate is high, flash a red light in my current location"*, will forward it to the middleware for the translation back to *"If my heart rate is above 100 bpm, make the smart light bulb in my living room flash red"*. Again, in order to trigger a flashing red light, we need a way to notify the smart home service. Since both IoT application use different rule formats, we need to investigate the best approach to handle this notification but one proposition is to let the smart health application notify the middleware, which in turn notifies the smart home application to trigger the light action.

The benefit of having the rules stored in a high-level format in a user's Solid pod also helps in fulfilling the user's requirement of being notified while on holiday. The flows described above remain the same for the user during their holiday. The applications at the holiday destination which have been granted access by the user can either execute the user's rules directly if they understand the high-level rule format or they can send the rules to the middleware for re-translation in order to execute them.

We also introduce the concept of a *user-managed run-time environment* in order to enable end-to-end ownership of IoT rules. With this concept, the user can opt to host an IoT application provided by a vendor themselves as indicated in Figure 1 with the private *IoT Smart Health service* and flows (6a) and (6b). This ensures that users do not just own their rules but also the environment in which these rules are executed.

5 CONCLUSIONS AND FUTURE WORK

We have identified two key problems in state-of-the-art IoT solutions, being *rule ownership* and *rule interoperability*. We demonstrated that with existing IoT platforms users lose the ownership of their rules and that the rules often written in a proprietary format are not transferable to different IoT platforms. As a step towards cross-platform IoT interaction, we proposed a cross-platform IoT architecture addressing the issues of rule ownership and rule interoperability via an IoT middleware as well as the use of Solid pods. Our proposed architecture allows each component to be developed independently, be decoupled from the users' data and be replaced at the will of the user. We are currently developing the proof-of-concept IoT middleware component introduced in Section 4 and expect to make various contribution in terms of a cross-platform IoT middleware, a user-managed run-time environment for the private execution of rules as well as a rich model for IoT rules.

We realise that our proposed solution might lead to a user's IoT solution being distributed across multiple IoT platforms. This in turn increases the complexity and intelligibility of implicit interactions in such a heterogeneous IoT environment. In this context, intelligibility calls for providing insights into a system's past and current states [2]. Rules that are possibly created with different tools and run on multiple platforms make it more difficult for a user to have an overview about which rules might possibly be in conflict with one another. We therefore also plan to further investigate intelligibility in the particular case of cross-platform IoT solutions.

REFERENCES

[1] Fulvio Corno, Luigi De Russis, and Alberto Monge Roffarello. 2019. A High-level Semantic Approach to End-User Development in the Internet of Things. *International Journal of Human-Computer Studies* 125 (2019), 41–54. https://doi.org/10.1016/j.ijhcs.2018.12.008

[2] Timo Jakobi, Gunnar Stevens, Nico Castelli, Corinna Ogonowski, Florian Schaub, Nils Vindice, Dave Randall, Peter Tolmie, and Volker Wulf. 2018. Evolving Needs in IoT Control and Accountability: A Longitudinal Study on Smart Home Intelligibility. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2, 4 (2018), 1–28. https://doi.org/10.1145/3287049

[3] Sylvain Kubler, Jérémy Robert, Ahmed Hefnawy, Kary Främling, Chantal Cherifi, and Abdelaziz Bouras. 2017. Open IoT Ecosystem for Sporting Event Management. *IEEE Access* 5 (2017), 7064–7079. https://doi.org/10.1109/ACCESS.2017.2692247

[4] Toby Jia-Jun Li, Yuanchun Li, Fanglin Chen, and Brad A Myers. 2017. Programming IoT Devices by Demonstration Using Mobile Apps. In *Proceedings of IS-EUD 2017, International Symposium on End User Development*. Springer, Eindhoven, The Netherlands, 3–17. https://doi.org/10.1007/978-3-319-58735-6_1

[5] Ken Peffers, Tuure Tuunanen, Marcus A Rothenberger, and Samir Chatterjee. 2007. A Design Science Research Methodology for Information Systems Research. *Journal of Management Information Systems* 24, 3 (2007), 45–77. https://doi.org/10.2753/MIS0742-1222240302

[6] Andrei Vlad Sambra, Essam Mansour, Sandro Hawke, Maged Zereba, Nicola Greco, Abdurrahman Ghanem, Dmitri Zagidulin, Ashraf Aboulnaga, and Tim Berners-Lee. 2016. *Solid: A Platform for Decentralized Social Applications Based on Linked Data*. Technical Report. MIT CSAIL & Qatar Computing Research Institute.

[7] Sergios Soursos, Ivana Podnar Žarko, Patrick Zwickl, Ivan Gojmerac, Giuseppe Bianchi, and Gino Carrozzo. 2016. Towards the Cross-Domain Interoperability of IoT Platforms. In *Proceedings of EuCNC 2016, European Conference on Networks and Communications*. IEEE, Athens, Greece, 398–402. https://doi.org/10.1109/EuCNC.2016.7561070

[8] Sandra Trullemans, Lars Van Holsbeeke, and Beat Signer. 2017. The Context Modelling Toolkit: A Unified Multi-layered Context Modelling Approach. *Proceedings of the ACM on Human-Computer Interaction* 1, EICS (2017), 1–16. https://doi.org/10.1145/3095810

[9] Blase Ur, Elyse McManus, Melwyn Pak Yong Ho, and Michael L Littman. 2014. Practical Trigger-Action Programming in the Smart Home. In *Proceedings of CHI 2014, ACM Conference on Human Factors in Computing Systems*. Association for Computing Machinery, Toronto, Canada, 803–812. https://doi.org/10.1145/2556288.2557420