sustainability

Article

Elis: An Open Platform for Mobile Energy Efficiency Services in Buildings

Paul Davidsson, Ulrik Eklund and Carl Magnus Olsson *

Internet of Things and People Research Center, Department of Computer Science and Media Technology, Malmö University, 205 06 Malmö, Sweden; paul.davidsson@mau.se (P.D.); ulrik.eklund@mau.se (U.E.)
* Correspondence: carl.magnus.olsson@mau.se; Tel.: +46-72-552-08-27

Received: 21 December 2018; Accepted: 2 February 2019; Published: 7 February 2019

Abstract: The recent years have witnessed an enormous growth of mobile services for energy management in buildings. However, these solutions are often proprietary, non-interoperable, and handle only a limited function, such as lighting, ventilation, or heating. To address these issues, we have developed an open platform that is an integrated energy management solution for buildings. It includes an ecosystem of mobile services and open APIs as well as protocols for the development of new services and products. Moreover, it has an adapter layer that enables the platform to interoperate with any building management system (BMS) or individual device. Thus, the platform makes it possible for third-party developers to produce mobile energy efficiency applications that will work independently of which BMS and devices are used in the building. To validate the platform, a number of services have been implemented and evaluated in existing buildings. This has been done in cooperation with energy companies and property owners, together with the residents and other users of the buildings. The platform, which we call Elis, has been made available as open source software under an MIT license.

Keywords: energy efficiency; building management; mobile services; internet of things; software platforms; open source

1. Introduction

The growing demand for energy efficiency is a global trend in many areas, not least in buildings. With increased urbanization, which in many countries coincides with a strong population growth, the energy demands in cities are difficult to meet unless action is taken [1]. Added to this are the well-known environmental problems of energy production and consumption. Politicians are well aware of this and therefore take various measures. For example, the city of Malmö, Sweden, has set a target to halve the energy consumption of its buildings by the year 2020. Since its inception in 2001, it has reached the halfway mark by taking simple measures such as additional insulation and timer control of lighting. However, moving forward requires more advanced solutions. Many other cities are in a similar situation. The existing energy efficiency potential in Swedish publicly-owned properties has been estimated to amount to SEK 3.7 billion (approximately USD 600 million) per year [2]. Obviously, there is a similar opportunity/challenge also for private property.

In the work presented here, we have developed a solution for how the intelligent use of ICT at the individual level can save energy in existing buildings. However, this is not its limitation. If a solution works in existing buildings, one can also implement it in new buildings, often even at a lower price. There is an intensive development in the energy efficiency of buildings, where new products are launched continuously. However, these solutions are often proprietary, non-interoperable, and handle only a limited range of functions, such as lighting, ventilation, or heating. To address future challenges and the interoperability of smart grids and micro-grids, as well as to make buildings and neighborhoods ready to interact with energy supply, solutions need to be multidisciplinary. In the future, more open energy markets will increase the need for decision support, to change energy provider, and the ability to commercially sell back energy, both on the M2M (machine-to-machine), the individual, and the organizational level.

To address these issues, we have developed an open platform that is an integrated energy management solution for buildings. It includes an ecosystem of services and open APIs and protocols for the development of new services and products. In our case, we have used mobile services for end-user validation, but non-mobile services may just as well be used. Our focus on mobile services was a result of the requirements of the end-users we had access to and what was relevant for them. Therefore, we maintain mobile services as our point of reference throughout this paper, even if the platform is not limited to this. Moreover, it has an adapter layer that enables the platform to interoperate with any building management system (BMS) or individual device. The purpose of the platform is not to replace BMSs but rather to

1. make it possible for third-party developers to produce mobile energy efficiency applications that will work independently of which BMS and devices are used in the building, and
2. integrate arbitrary BMSs and other relevant systems and devices in the building so that these can be seen as one system from the application developer's and the user's perspective.

To validate the platform, a number of services have been implemented and evaluated in existing buildings. This has been done in cooperation with energy companies such as E.On and Schneider Electric; ICT companies such as Ericsson, Sony, and IBM; property owners such as the City of Malmö; and together with the residents and other users of the buildings. The platform, which we call Elis, has been made available as open source under an MIT license. E.On and Schneider Electric supplied the project with knowledge on and access to BMS solutions, while the ICT companies provided feedback on mobile service design, expert reviews of architecture, and hands-on code inspection. The City of Malmö provided us with a suitable school for the co-development of end-user services and access to the BMS and through the municipality property owner MKB we were granted access to apartment owners willing to have their apartments retrofitted and take part in the study.

In the next section, we provide a review of some related work. We then describe the platform and the main issues that were considered during the different stages of its development. This is followed by the validation of the platform and some concluding remarks.

Related Work

There is a substantial amount of research and development performed in the area of using modern ICT for energy efficiency in buildings. Ref. [3] provide an extensive review of such work including a list of functional and non-functional requirements for BMSs. Another comprehensive review is provided by [4], where different aspects are discussed, including context-sensing, activity and context inference, and energy-aware service-based middleware, etc. A number of reviews have been conducted but none for platforms/systems which provide mobile services to users. These include a review of advanced control systems engineering for energy management [5]; the design, management, and operation of intelligent buildings [6]; of ICT for energy efficiency [7]; of context-aware tools for smart home development [8]; of smart homes and future challenges [9]; of smart environments [10]; of user activities in energy intelligent buildings [11]; and of measuring and evaluating intelligent buildings and their sustainability [12].

From our analysis of research describing existing ICT systems and platforms for energy efficiency in buildings, we conclude that they typically

- do not support the use of mobile applications (for smartphones)
- have a "closed" system architecture in at least two ways:
    - do not support arbitrary BMSs and devices
    - do not provide an open API for third party developers
- have not been evaluated in real-life settings.

It is further noteworthy that commercial home automation platforms such as Amazon Echo, Apple HomeKit, and Google Home, as well as open source alternatives like Calaos, Domoticz, and MisterHouse, do not compare with the type of platform discussed in this paper as they (1) are for home automation only, (2) serve a convenience purpose rather than hold a thorough focus on energy efficiency, and (3) emphasize integration of smaller individual devices rather than full BMSs.

2. Materials and Methods

Methodologically, this study relies on the well-defined steps within design research [13]. Step one is the motivation for the study, and, in our case, it is triggered by the gap in the existing research identified. Ref. [13] describe such gaps as a natural starting point for design research with objective-centered solutions. This typically results in the development of an artifact that strives to explore hitherto unaddressed solutions by including knowledge from the problem domain and current solutions.

Step two is the design and development of our platform. This corresponds with the argument by [13] for including new properties in such artifact studies, such as the desired functionality and architecture for the identified gap.

In the third step, we provide three service implementation examples to demonstrate how the artifact contributes to reaching our objective of making it possible for third party developers to produce mobile applications independently of which BMSs and devices are used in a building.

Finally, the validity of our evaluation strategy in step four is in line with [14]. We support this by providing an observational qualitative evaluation, an analytical evaluation, and a descriptive evaluation.

3. Results

Before describing the resulting platform, we will discuss the basic requirements and quality attributes considered, as well as the main design choices made as a consequence of this.

3.1. Main Design Choices

As simplicity and interoperability of the public API were priorities, we adopted the representational state transfer (REST) software architecture style [15]. REST consists of guidelines and best practices for creating scalable web services. As a consequence, we had to accept that not all proprietary services of, e.g., BMSs and devices are supported and made available to third party developers.

As modifiability, scalability, and cost for infrastructure are important aspects, we followed the OSGi specification [16] for implementing the platform. As a consequence, we had to accept that this means translating device and vendor services to and from Java within the platform.

Moreover, to further minimize cost, it is always assumed that it must be possible to use existing infrastructure and devices, and to support openness, the platform has been made available as open source under an MIT license at GitHub [17] (see Supplementary Materials).

Finally, to further support usability, interoperability, and scalability, the architecture adheres to the IoT-A [18] architectural reference model (ARM), where applicable.

3.2. The Platform Architecture

The architecture of the Elis platform can be described according to different views. We will begin with the logical view.

3.2.1. The Logical View

The main role of the platform is to serve as middleware between the BMS, device services, external web services, etc., and the (generic) mobile service applications. The platform contains a number of internal services that provide support for, e.g., monitoring, data management, and control. Figure 1 shows the design dependencies from the platform to the proprietary building and house management systems and down to the physical devices included in the latter systems.

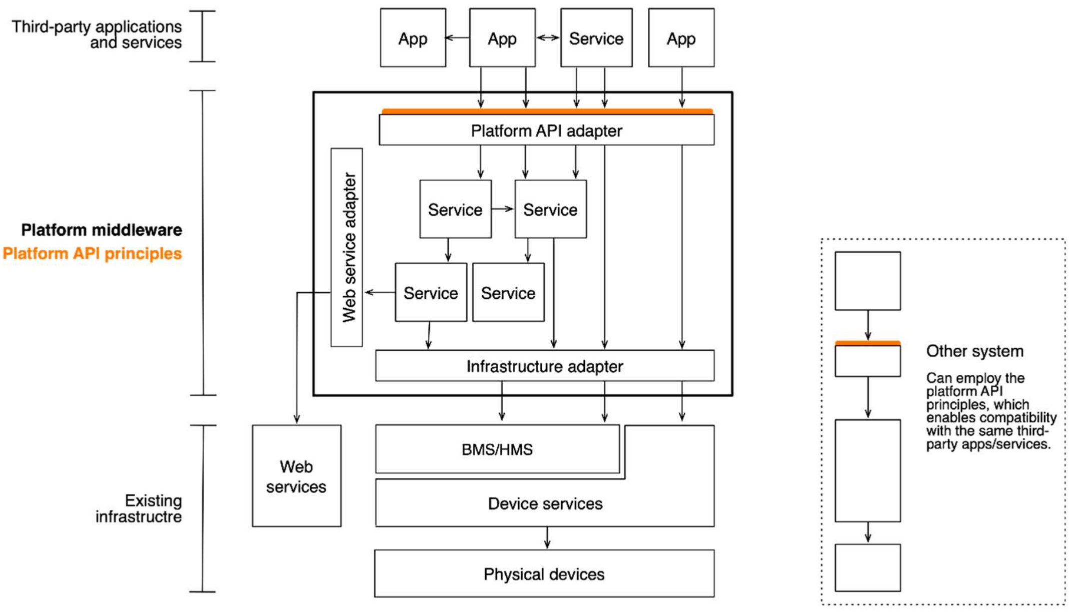


Figure 1. Logical view showing the design dependencies of the platform to the proprietary systems.

The key aspect for third-party developers is the consistency of the open API (the red line) over different systems, enabled by the platform API adapter layer. The key aspect for developing the Elis platform, on the other hand, is the infrastructure adapter layer. A third type of adapter, the web service adapter, allows platform services access to web services outside the building system, e.g., weather services. These adapter layers and the internal services of the platform are developed within the OSGi framework, which supports modifiability, scalability, and independence from the hardware on which the platform runs.

3.2.2. The Deployment Configuration View

On the right in Figure 2, the general deployment configuration for resources, services, and users is shown. In this case, the users can be, e.g., a web application or a mobile app. The HTTP API used between the user and the service conforms with REST principles level 2 of the REST maturity model [19].

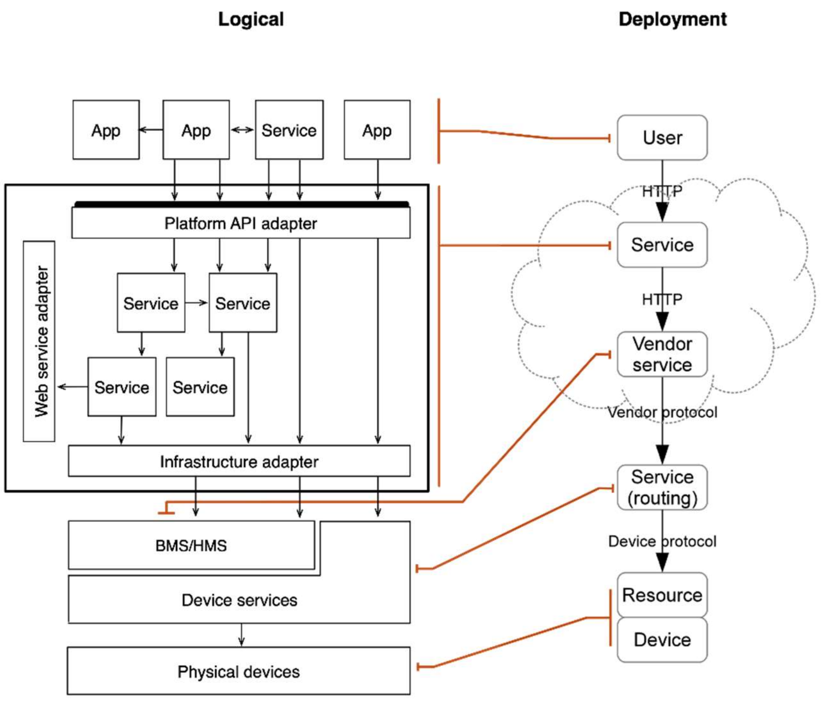


Figure 2. Deployment of the structural elements to various types of running software and how they interact.

Figure 2 also shows the unequivocal deployment of the logical elements of the platform and how they are deployed to running software. The platform services are shown as cloud deployed to indicate the independence from the hardware on which it is executed.

3.2.3. The Physical View

As illustrated in Figure 3, a number of electrical devices in the building are typically connected (e.g., via smart-plugs) to a local gateway that consists of a communication function and possibly a local server with processing and storage capabilities. The gateway enables IP-based communication with external units. The local server may include intelligence useful for implementing energy management services. This intelligence and the data storage could also reside in a remote cloud server.

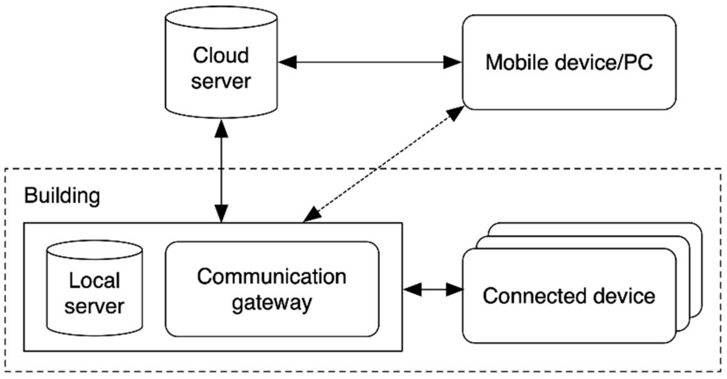


Figure 3. General physical layout.

The users interact with the system via a personal computer or a mobile device, e.g., a smartphone or a tablet computer, which then communicates with either the cloud server or the communication gateway in the building. The intelligence and data storage may also reside on the PC or mobile device.

There are at least three different ways the deployment configuration elements are allocated to a physical system, the main difference being how the vendor and platform services are deployed; either locally to the building/home or deployed to a cloud server. In Figures 4–6, these alternatives are illustrated.

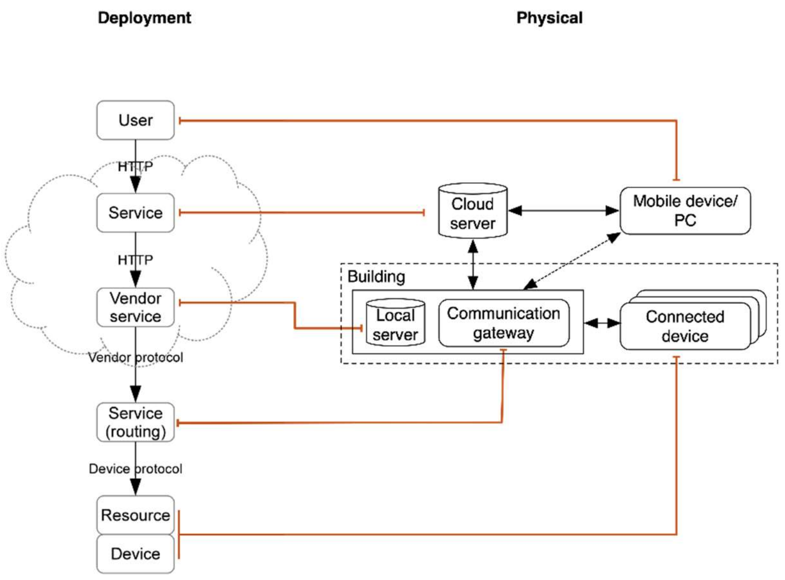


Figure 4. Deployment according to which the Elis platform services run on a separate cloud server.

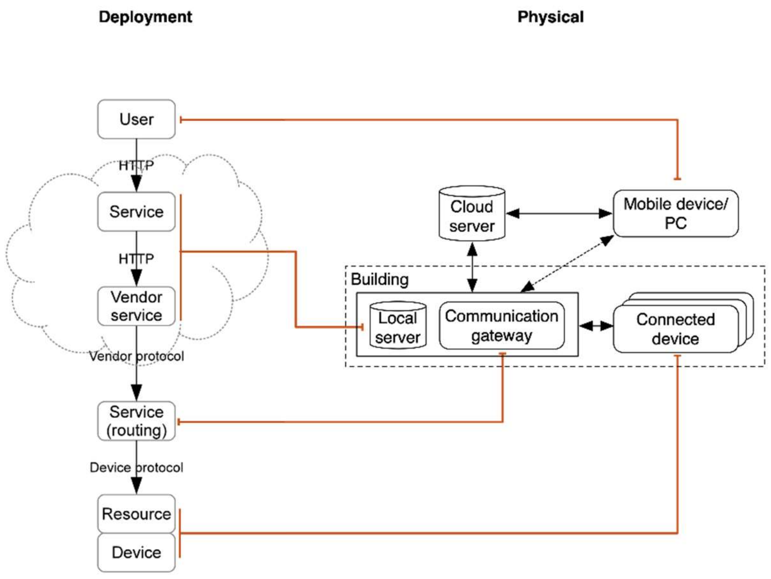


Figure 5. Deployment according to which both the vendor services and the Elis platform services run on a local server within the building system.

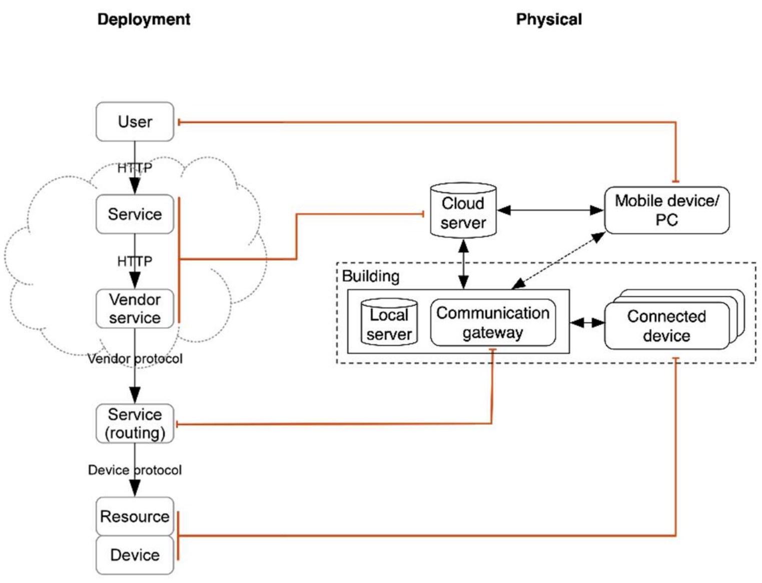


Figure 6. Deployment according to which both the vendor services and the Elis platform services run on a cloud server.

3.3. Platform Services

The Elis platform and its internal services can be represented in a model of successive shells encompassing each other (see Figure 7). These shells represent two things:

- the stages of how the system is built; i.e., each shell (including elements inside) can be a standalone system not depending on the outer shells.
- the levels of abstraction between the services and also what services are fundamental to running the platform.

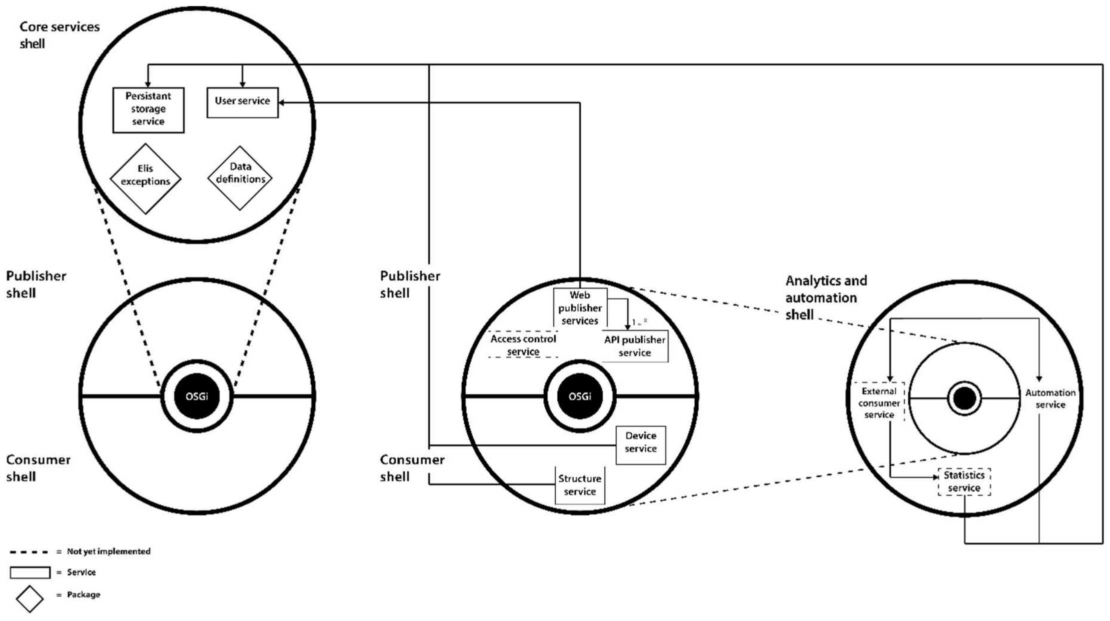


Figure 7. The shell model for platform services.

The services in the core services layer represent the lowest level of the Elis platform, that is, the services and packages that are required to instantiate the platform. The services in this layer are:

- persistent storage service: handles persistence in the platform.
- user service: handles the mapping of gateway users to platform users.
- Elis exceptions: handle all the custom exceptions in the platform.
- data definitions: handle repurposing of data sets. This could, for example, include handling temperature, energy pricing, etc.

Outside the core services shell is the publisher/consumer shell. This shell is divided into two areas: the top half, called the publisher shell, publishes functionality in the platform through an API. The lower half, the consumer shell, consumes services from e.g., a BMS and repurposes information from such services in another API.

The services in the publisher shell are:

- web publisher services: handle third-party developers in the platform.
- API publisher service: publishes services as resources for third-party developers through an API.
- access control service: handles all the role-based access in the platform (not implemented yet).

The services in the consumer shell are:

- device service: maps capabilities from the various BMSs to the Elis platform. It effectively describes the capabilities of each BMS and its connected peripherals.
- structure service: used to represent the structure of a physical building, i.e., rooms, apartments, etc.

Outside the publisher/consumer shell is the analytics and automation shell. It contains the following services, which both utilize and modify existing data, but also receive new data from external sources:

- external consumer service: fetches information from external service providers, for example, weather information from weather APIs or current energy prices from, e.g., Nord Pool Group [20] (not implemented yet).
- statistics service: utilizes data in the platform to aggregate data and create statistics (not implemented yet).
- automation service: is used to modify/change state on devices connected to a BMS based upon input. This input can be either a third-party application requesting to turn off a light at a specific time during the day or through monitoring a change in value from an external provider (not implemented yet).

3.4. The Process View

The process, or run-time, view is constituted of a set of scenarios that detail the system behavior. We here describe the most important scenario through a sequence diagram with elements from the deployment configuration view.

The scenario concerns how a proprietary device in a BMS for homes, or home management system (HMS), is controlled. This includes the handling of timeouts and other errors. Errors in the system are usually seen as delays or omitted responses from the devices. The scenario in Figure 8 illustrates how the platform acts when an error occurs while controlling a device. The process may differ somewhat depending on the underlying service as each service may hold different assumptions, some allowing for subscriptions to updates while others may expect acknowledgments to be returned. Figure 8 is based on an E.On HMS, which assumes that a request from the platform to, for example, turn on a light is something that the platform continuously asks for until the HMS replies that it has been done.

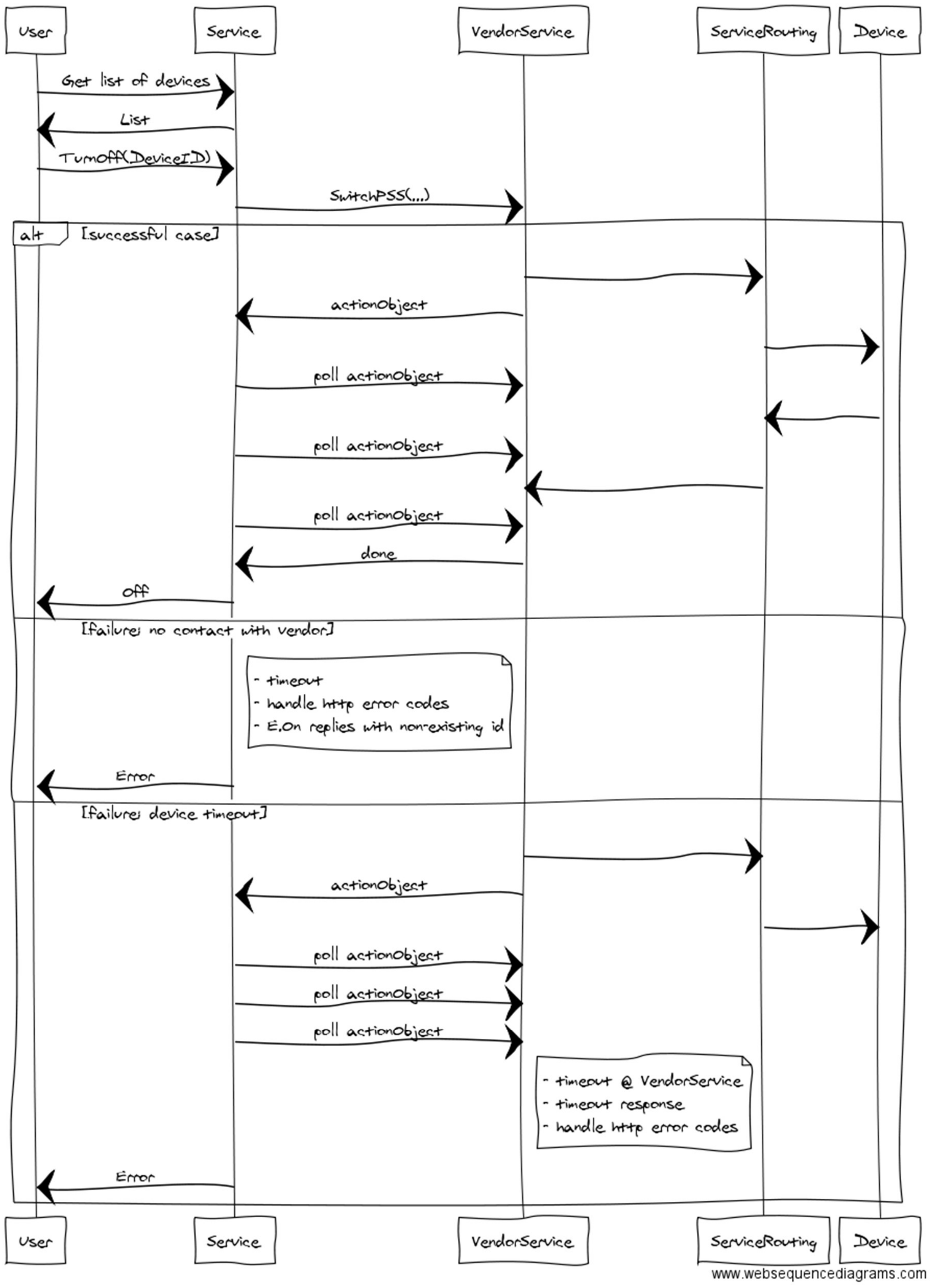


Figure 8. Controlling a device in an E.On HMS, handling timeouts and other errors.

The error reply could contain where the error happens, i.e., how far did the request get. Inconsistencies between the user and the platform may be resolved by a device refresh.

4. Validation and Discussion

To validate the Elis platform, we have implemented a number of mobile services for energy efficiency as well as a set of adapters to BMS/HMS and device services. Moreover, the energy efficiency services have been developed and tested in a living labs–based setting [21].

4.1. Services Implemented

4.1.1. Elis Crowdis

This prototype investigates services specifically targeted to public buildings, which are represented by schools in the Elis project. One of the services is concerned with how, through crowdsourcing, it is possible to allow the people currently in the building to vote on issues that are directly related to the current energy consumption. In the prototype, we focused on the state of the classroom in terms of airflow, air quality, or temperature. During testing of the first prototype it was realized that the service should limit voting on room conditions to people that were currently in the room. As Figure 9 shows, we used web-based geolocation to determine the current location of the device and then displayed voting options if the device was within a certain area, also known as a geofence.

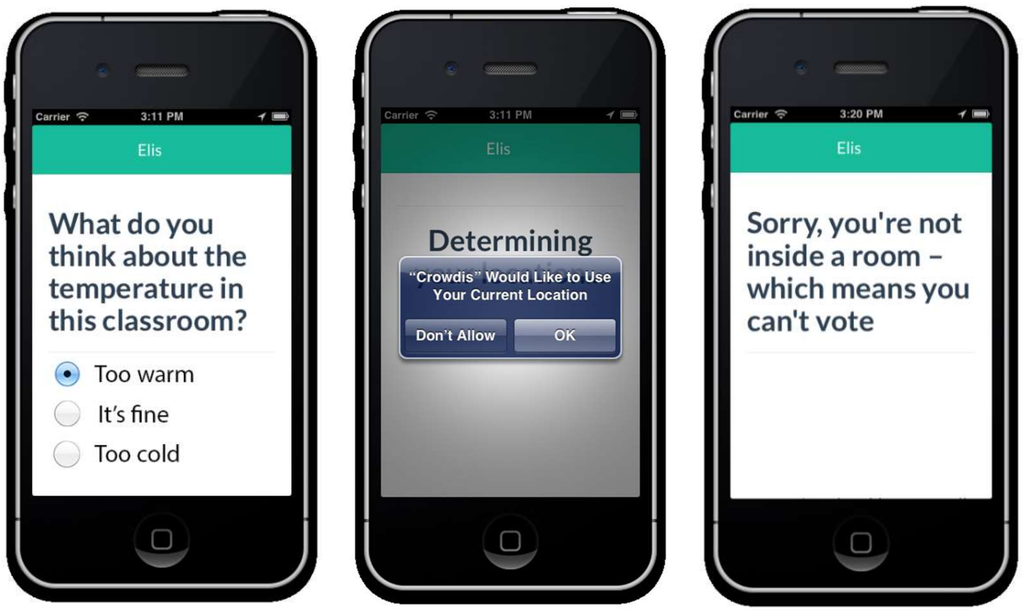
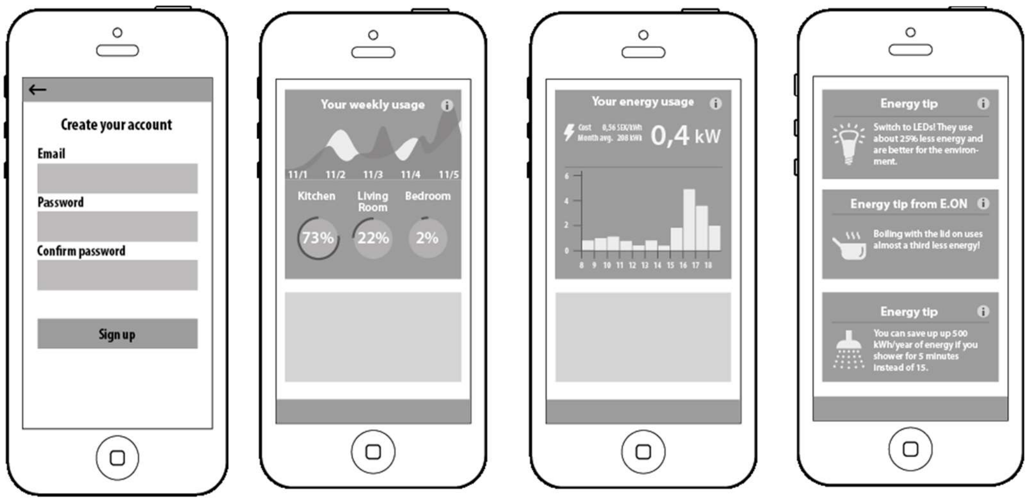

Carrier
3:11 PM
Elis
What do you think about the temperature in this classroom?
Too warm
It's fine
Too cold
Carrier
3:11 PM
Elis
Determining
"Crowdis" Would Like to Use Your Current Location
Don't Allow
OK
Carrier
3:20 PM
Elis
Sorry, you're not inside a room – which means you can't vote

Figure 9. The Elis Crowdis prototype.

4.1.2. Elis Mobile

Elis Mobile is the main prototype that was designed and evaluated with users. Elis Mobile aims to make users understand their energy usage and how their behavior affects energy consumption. The design paradigm used is that of cards (see Figure 10). This was inspired by the Google Now card flow found in their mobile application for both iOS and Android. The paradigm was tested through paper prototyping with users and the prototype was developed based on this input. The prototype was built using the Ionic framework and AngularJS in unison with Node.js to create an application on only web-based technologies (HTML5).



Switch to LEDs! They use about 25% less energy and are better for the environment.

Boiling with the lid on uses almost a third less energy!

You can save up up 500 kWh/year of energy if you shower for 5 minutes instead of 15.

Figure 10. The Elis Mobile prototype.

4.1.3. The ELIS Game

In terms of game mechanics, this is a simple Flappy Bird [22] clone. It is built using HTML5 and runs on mobile devices as well. What sets this apart from a regular Flappy Birds clone game is that the mechanics can be manipulated in real-time and that the difficulty of the game depends on the current energy consumption. A web interface makes it possible for the game administrator to change values that directly influence the gameplay for the player(s). This is done through three-way data binding using AngularJS and Firebase.io for database storage of the values across multiple clients. This interface also has a setting that can set the values automatically. The way this is done is through fetching the current energy consumption for a specific user's apartment through the Elis publisher API. The value fetched from the API is then used to set the level of difficulty in the game. This is done every 5 seconds or until the admin turns off the automatic settings.

4.2. Core Requirements Validation

In relation to the list of suitable evaluation forms for practical design-oriented research outlined by [14], we followed one or more of these strategies:

1. An observational qualitative evaluation through a number of limited case studies; we studied the platform in the intended business environment, however, in some cases with some artificial constraints compared to a commercial system.
2. An analytical evaluation through static analysis where we examined the structure of the platform for static qualities, e.g., complexity. We also studied the fit of the platform into the technical architecture together with other systems.
3. A descriptive evaluation through an informed argument: We used information based on our knowledge base, e.g., relevant research, to help building the argument for the platform's properties.

The quality attribute of openness of was demonstrated by the ease of implementation of the mobile services described above, i.e., strategy 1. It was possible for a single programmer to implement these to a status usable for beta-testing with end-users that allowed this programmer to focus on front-end design and experiences from users rather than platform integration and data access. The API in the platform adapter was also analyzed (strategy 2 above) to establish to what level of conformity to REST it had, which is considered an established standard to support openness in web APIs [23].

Interoperability was demonstrated by the number of diverse architectures for BMSs that the infrastructure adapter supports, thus demonstrating the property by using strategy 1. Specifically, in order to validate the proposed adapter approach, the following concrete adapters were implemented:

E.On Smart Home HMS, Ninja Blocks, Schneider Electric Enterprise Server, Sensegate by Sigma Connectivity, Philips Hue, and Nexa Sunrise.

Modifiability was demonstrated by evaluation strategy 1 throughout the project lifetime of 2 years, with the platform being extended with additional services based on discussions with various project stakeholders. We were also able to see that design of the platform architecture in different shells also helped in confining changes when adding new services.

We have not demonstrated security and privacy to the desired extent since all necessary services for this had not been implemented in the platform. We made provisions for where these services should be implemented (see access control in the publisher shell). Although the access control service has not yet been implemented, a thorough risk exposure and security analysis of the Elis platform has been carried out [24]. It was concluded that out of 32 examined risks, 9 were classified as low and 4 as high; that is, most of the identified risks were considered moderate. The risks classified as high were either related to the human factor or to software. Moreover, the results indicated that with the implementation of standard security features, most of the risks could be minimized to acceptable levels. However, the most serious risks, i.e., those derived from the human factor, need more careful consideration as they are inherently complex in nature.

The main cost of a smart home automation system based on the Elis platform comes from the physical devices and connecting them to the servers running the Elis platform. The platform itself was demonstrated to be moved easily to the servers at Malmö University IT department with minimal server cost. The platform was also easy to move between servers, due to the use of OSGi, which also minimized the operational costs in the project.

The usability of the platform API was tested through prototype development of external services. For example, the Flappy Elis game relied upon the simplicity of the API in order to allocate the majority of development time to the gamification aspects of the application. Prototypes were developed and tested in a living labs setting [21] and were thus validated through strategy 1.

The testing of scalability as well as performance was done from a pragmatic perspective, relying on user testing to report any issues. Thus, these two quality attributes should be assessed for any further end-user services as well as BMS/HMS integrations to ensure that performance meets the needs of each such scenario. Our reliance on strategy 1 for validation should thus be complemented in future work.

5. Conclusions and Future Opportunities

Current mobile services for the energy management in buildings are often proprietary, non-interoperable, and handle only a limited range of functions, such as lighting, ventilation, or heating. To address these issues, we have developed an open platform called Elis that includes an ecosystem of mobile services and open APIs as well as protocols for the development of new services and products. Moreover, it has an adapter layer that enables the platform to interoperate with any building management system or individual device.

The Elis platform makes it possible for third-party actors to develop mobile energy efficiency applications that will work independently of which BMSs and devices are used in the building. We have validated the openness, interoperability, modifiability, and usability of the platform, through the implementation of a number of services and applying them in existing buildings. This validation was done in cooperation with energy companies and property owners, together with the residents and other users of the buildings.

The Elis platform is available as open source under an MIT license, and we welcome involvement and extensions in accordance with this license. Future work may, e.g., include further validation of scalability and performance, implementing security mechanisms, and developing the built-in analytics and automation services. In the version of the platform available as open source, the product-specific dependencies and adaptations that were made during the project have been removed. The platform is a minimal viable product in order to promote flexibility for deployment and the integration with existing BMSs and as described in the provided documentation. Further steps to develop the platform into a commercial product are not intended as part of our research, which is why we have selected MIT as license, since it gives particularly generous means for further development to those with such interests.

We see three main uses of the platform in its publicly available version. First, it could be used in its entirety, which implies integrating it with one or several existing building and/or home management systems by developing an adapter according to the platform specification and developing end-user services based on the open API that the platform defines. Second, it could be used in parts, which implies that selected elements of the platform are used as components in another platform—potentially even copyrighted closed-source, as this is part of why an MIT license was used. For the project partners, this approach was key, as several of our collaborating organizations have their own building and/or home management systems but may for instance see particular value in the API for third-party developers, which follows consistent and well-documented state-of-the-art principles. Finally, the platform may be used as point of reference, such as in decision making processes for organizations that are considering developing their own platform. As platform development brings considerable risk for lock-in to early design choices, having an openly available platform that strives to integrate services and devices from any product ecosystem may serve as a useful starting point for risk assessment and complexity reviews. Our mapping towards the IoT-A reference architecture (available in the open source documentation) also provides an illustration of how to relate to such architecture descriptions, as they in themselves are not intended to be used in full but rather to aid in the decision making of what to focus on and why in specific implementations.

Supplementary Materials: The platform source code and documentation has been made available as open source under an MIT license at GitHub [17].

Author Contributions: Conceptualization: P.D., U.E., and C.M.O.; data curation: P.D., U.E., and C.M.O.; formal analysis: P.D., U.E., and C.M.O.; funding acquisition: P.D.; investigation, P.D., U.E., and C.M.O.; methodology: U.E., and C.M.O.; project administration: C.M.O.; resources: P.D., U.E., and C.M.O.; supervision: P.D. and C.M.O.; validation: P.D., U.E., and C.M.O.; visualization: U.E. and C.M.O.; writing—original draft: P.D., U.E., and C.M.O.; writing—reviewing and editing: P.D. and C.M.O.

Funding: This research was funded by the Knowledge Foundation through the project "Internet of Things and People Research Profile" and by VINNOVA through the project "Mobile services for energy efficiency in existing buildings" (2012-01245).

Acknowledgments: We acknowledge the valuable contributions from Marcus Ljungblad, Johan Holmberg, Axel Olsson, Joakim Lithell, and Fredrik Ohlin in the software development of the Elis platform and services.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Li, C.; Song, Y.; Kaza, N. Urban form and household electricity consumption: A multilevel study. *Energy Build.* **2018**, *158*, 181–193. [CrossRef]
2. Schneider Electric Report on mynewsdesk.com: 3,7 Miljarder Kronor per år, 2011. Available online: http://www.mynewsdesk.com/se/schneider-electric/documents/rapport-3-7-miljarder-12812 (accessed on 21 December 2018).
3. De Paola, A.; Ortolani, M.; Lo Re, G.; Anastasi, G.; Das, S.K. Intelligent management systems for energy efficiency in buildings: A survey. *ACM Comput. Surv. (CSUR)* **2014**, *47*, 13–38. [CrossRef]
4. Amft, O. D1.1b: State of the Art Review, IST EU STREP Project FP7-258888. 2012. Available online: http://www.greenerbuildings.eu/sites/greenerbuildings.eu/files/main.pdf (accessed on 21 December 2018).
5. Dounis, A.I.; Caraiscos, C. Advanced control systems engineering for energy and comfort management in a building environment—A review. *Renew. Sustain. Energy Rev.* **2009**, *13*, 1246–1261. [CrossRef]
6. Clements-Croome, D.J. (Ed.) *Intelligent Buildings: Design, Management and Operation*, 2nd ed.; ICE Publishing: London, UK, 2013; pp. 1–304.

7. Ye, J.; Hassan, T.M.; Carter, C.D.; Zarli, A. ICT for Energy Efficiency: The Case for Smart Buildings, Department of Civil and Building Engineering, Loughborough University, 2008. Available online: http://itc.scix.net/data/works/att/w78-2009-1-66.pdf (accessed on 21 December 2018).
8. Robles, R.J.; Kim, T. Review: Context Aware Tools for Smart Home Development. *Int. J. Smart Home (IJSH)* **2010**, *4*, 1–12.
9. Chan, M.; Estève, D.; Escriba, C.; Campo, E. A review of smart homes—Present state and future challenges. *Comput. Methods Programs Biomed.* **2008**, *91*, 55–81. [CrossRef] [PubMed]
10. Cook, D.J.; Das, S.K. How smart are our environments? An updated look at the state of the art. *Pervasive Mob. Comput.* **2007**, *3*, 53–73. [CrossRef]
11. Nguyen, T.A.; Aiello, M. Energy intelligent buildings based on user activity: A survey. *Energy Build.* **2013**, *56*, 244–257. [CrossRef]
12. Alwaer, H.; Clements-Croome, D.J. Key performance indicators (KPIs) and priority setting in using the multi-attribute approach for assessing sustainable intelligent buildings. *Build. Environ.* **2010**, *45*, 799–807. [CrossRef]
13. Peffers, K.; Tuunanen, T.; Rothenberger, M.A.; Chatterjee, S. A Design Science Research Methodology for Information Systems Research. *J. Manag. Inf. Syst.* **2007**, *24*, 45–77. [CrossRef]
14. Hevner, A.R.; March, S.T.; Park, J.; Ram, S. Design Science in Information Systems Research. *MIS Q.* **2014**, *28*, 75–105. [CrossRef]
15. Fielding, R.T.; Taylor, R.N. Principled design of the modern Web architecture. In Proceedings of the 22nd International Conference on Software Engineering, Limerick, Ireland, 4–11 June 2000; ACM: New York, NY, USA, 2000; pp. 407–416. [CrossRef]
16. OSGi Alliance Specifications. Available online: http://www.osgi.org/Specifications (accessed on 6 February 2019).
17. GitHub Elis Platform. Available online: https://github.com/iotap-center/elis-platform (accessed on 6 February 2019).
18. Carrez, F.; Bauer, M.; Boussad, M.; Bui, N.; Jardak, C.; De Loof, J.; Magerkurth, C.; Meissner, S.; Nettsträter, A.; Olivereau, A.; et al. Internet of Things—Architecture IoT-A, Deliverable D1.5—Final Architectural Reference Model for the IoT v3.0. European Union, 7th Framework Programme. 2013. Available online: https://www.researchgate.net/publication/272814818_Internet_of_Things_-_Architecture_IoT-A_Deliverable_D15_-_Final_architectural_reference_model_for_the_IoT_v30 (accessed on 6 February 2019).
19. Richardson Maturity Model. Available online: http://martinfowler.com/articles/richardsonMaturityModel.html (accessed on 6 February 2019).
20. Nord Pool Group. Available online: https://www.nordpoolgroup.com (accessed on 6 February 2019).
21. Niitamo, V.P.; Kulkki, S.; Eriksson, M.; Hribernik, K.A. State-of-the-art and good practice in the field of living labs. In *Proceedings of IEEE International Technology Management Conference*; IEEE: Milan, Italy, 2006; pp. 1–8. [CrossRef]
22. Flappy Bird. Available online: https://en.wikipedia.org/wiki/Flappy_Bird (accessed on 6 February 2019).
23. Mäkeläinen, S.; Alakoski, T. Fixed-mobile hybrid mashups: Applying the rest principles to mobile-specific resources. In *International Conference on Web Information Systems Engineering*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 172–182.
24. Jacobsson, A.; Boldt, M.; Carlsson, B. A Risk Analysis of a Smart Home Automation Systems. *J. Future Future Gener. Comput. Syst.* **2016**, *56*, 719–733. [CrossRef]