R. Reussner, A. Koziolek, R. Heinrich (Hrsg.): INFORMATIK 2020,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2021 979

Requirements and Mechanisms for Smart Home Updates

Peter Zdankin[1] Oskar Carl[2] Marian Waltereit[3] Viktor Matkovic[4] Torben Weis[5]

Abstract: The interconnection of sensors and actuators of smart home devices creates dependencies that allow for ubiquitous services. These devices can be subject to transformative changes through software updates that might lead to unintended consequences. Users have no tools to predict the negative consequences caused by updating their smart home. In this paper, we address this problem and propose mechanisms that enable organized update planning in a smart home. We compare self-description standard approaches that allow reasoning about resulting functionality before updates are installed. Updating devices to their latest versions is not necessarily the best way to update smart homes, therefore we discuss multi-objective optimization in the update process. Finally, outsourcing functionality to external providers might reduce the complexity of certain tasks, but can also pose threats if the wrong tasks are offloaded.

Keywords: Smart Home; Longevity; Self-Description; Update Configuration; Edge Computing

1 Introduction

A smart home is composed of multiples devices from various vendors. Each vendor is producing software updates on his own, which leaves it to the user to select the best set of updates. Vendors cannot always consider all possible setups, and therefore faults can occur when a device is updated. Furthermore, downgrading software versions is not always possible, which means that a single unfortunate update can permanently impact the functionality of a smart home. Currently, users have no tools to master the update problem without risking damage to the system. Thus, it is important to investigate the update process and to propose solutions to compute the optimal update configuration automatically. In Section 3 we discuss how self-description of devices and services can principally solve the update problem. Then we discuss possible definitions of optimality of such a solution in Section 4. Finally, in Section 5 we compare solutions to the update problem relying on centralized services in the cloud with solutions that work locally in the smart home and discuss their impacts on ease of use and autonomy.

[1] University of Duisburg-Essen, Distributed Systems Group, Duisburg, Germany, peter.zdankin@uni-due.de
[2] University of Duisburg-Essen, Distributed Systems Group, Duisburg, Germany, oskar.carl@uni-due.de
[3] University of Duisburg-Essen, Distributed Systems Group, Duisburg, Germany, marian.waltereit@uni-due.de
[4] University of Duisburg-Essen, Distributed Systems Group, Duisburg, Germany, viktor.matkovic@uni-due.de
[5] University of Duisburg-Essen, Distributed Systems Group, Duisburg, Germany, torben.weis@uni-due.de

doi:10.18420/inf2020_91

2 System Model

The architecture of smart home systems may vary considerably between different implementations. Some systems are designed and deployed during the construction of a house and permanently deployed e.g. via bus systems in walls. While this is an interesting option, most homes require expensive renovation to become smart this way. An alternative to this is the usage of modular subsystems, for example, standalone lighting or heating systems, which can be bought individually and may offer their own separate platform, or integrate into bigger smart home platforms such as Amazon Alexa, Google Home, Samsung SmartThings or HomeKit. The benefit of modular smart home systems is a lower entry price for consumers, as they might start off with light bulbs or thermometers that can be controlled through already available smartphones or affordable hubs. In this paper, we focus on the latter, as the distributed and heterogeneous nature of these systems is more likely to break at some point during its lifetime, in comparison to a permanent solution installed during house construction. A smart home configuration can be described through the following system model:
*A **smart home** has a set of **devices** that are connected through a **platform**. Each device has a certain **software version** and a set of available **updates**. Each software version has a set of available **predefined services**. Devices can use services of other devices which creates **dependencies**. An **update configuration** is one of the finite states of the nondeterministic finite automaton NFA that can be constructed by using the configurations as states and connecting them using individual updates as transitions.*

During the lifetime of devices in a smart home, security issues or new features might require new software, which might alter the functionality of a device through for example a modified public API. These alterations might be intended, as part of a necessary change, or accidental because certain side effects were not considered. In any way, if such an update is installed, it will likely break an existing dependency. Currently, users cannot predict the changes that will be imposed on a smart home if a subset of its devices is updated simultaneously. Naive approaches that only consider individual updates for each update step, miss the broad picture because dependency problems can manifest after a certain set of devices have been updated already and rolling them back to previous versions may not always be possible.

3 Self-Description of Smart Home Devices

To enable interoperability, devices can describe their services to other devices, as was proposed in an architecture by Barbas et al. [Ve12]. Devices must be able to list all their currently available services, either through self-awareness or information included in the updates. By comparing the currently available services with the available services of an update, the differences can be computed. That way a system can predict possible faults in dependencies before they occur. However, comparing service definitions cannot capture unintended incompatibilities due to programming errors. Furthermore, most service

definitions are only syntactic, e.g. via an API definition, but they do not specify behaviour. Thus, an update might cause a change of behaviour without a change of its service definition. Two widely used approaches for service definition are *descriptive* or *prescriptive* standards.

3.1 Descriptive vs Prescriptive

A standard for service definitions that only regulates how a device can describe itself is a *descriptive standard*, because it allows vendors to describe their devices in their own terms. A recently finalized descriptive standard is the Web Of Things [Ka20; Ko20]. If devices need to communicate across different descriptive standards, a translation must be considered. However, McCool et al. have stated security concerns about purely descriptive approaches, such as scanning for door locks with known physical weaknesses [MR18].

A *prescriptive* standard does not only regulate the *how* but also the *what* of self-description. These standards prescribe how devices and services should be defined and vendors can use these terms for unambiguity. By using the same terms to define services, translation is not required anymore. A prescriptive standard defines all possible devices and all their services they could implement. However, a device can implement a subset of these services only. This restriction has benefits as well because it ensures compatibility across device vendors. Widely used examples are the smart home standards of Amazon, Apple, and Samsung[6].

3.2 Descriptive Standards Only

In a descriptive standard, each vendor can create its own namespace and definition for a certain type of device. This open approach is flexible and allows vendors to act quickly and independently. It also allows vendors to reinvent service definitions, as they are incentivized to invent proprietary definitions in order to support new features. This can lead to multiple definitions of the same device type, thus creating redundancy as no incentive is given to find an agreement between vendors. As multiple definitions can coexist, communicating devices might need to translate between the definitions. Alternatively, a middleware could be introduced to translate between incompatible but equivalent descriptions. Over time, device vendors could update their own definitions in ways that impair interoperability, even to the point of defective device functionality. Therefore, we assume that descriptive standards alone are not suited for the evolving smart home ecosystem.

[6] Amazon Alexa Skill API Documentation: `https://developer.amazon.com/de/docs/smarthome/understand-the-smart-home-skill-api.html` (accessed May 6, 2020),
Samsung SmartThings: `https://smartthings.developer.samsung.com` (accessed May 6, 2020),
Apple HomeKit Accessory Protocol Specification: `https://developer.apple.com/support/homekit-accessory-protocol/` (accessed May 6, 2020)

3.3 One Prescriptive Standard

In a single prescriptive standard, each device type and service would be defined free of redundancy. While this allows for optimal interoperability, many practical problems require consideration. First of all, a regulatory body needs to be decided on to define this standard. This might cause long delays in the process of reaching consensus between the stakeholders, given the current interest in smart home applications.

If only devices that are defined in this standard can function in such a smart home, innovative new products that are not yet defined must also first pass the committee before being able to operate in a smart home. This does not just create obstacles for new products, it also enables competitors in the market to start production of these devices while they are being standardized. Small vendors that invent new smart home products might have problems to compete against bigger ones, due to the time loss introduced by the certification process. Due to this, we assume that a single prescriptive standard is too restrictive since the domain is evolving rapidly.

3.4 Hybrid Standards

Based on the above conclusions, we propose a hybrid solution between descriptive and prescriptive standards. Smart home devices are usually connected to a single platform and not part of multiple platforms simultaneously. Hence, a prescriptive standard per platform could have the desired flexibility, because platform vendors can innovate independently. To improve interoperability, this can be combined with a descriptive standard between all platforms. All devices – regardless of their respective vendors and platforms – would self-describe using that descriptive standard but adhere to the rules of the prescriptive standard given by the respective platform vendor. This way, each platform maintains optimal interoperability via prescriptive definitions. Since all prescriptive definitions are instances of a single descriptive language, it is still possible to translate between multiple platforms as discussed above. Furthermore, it is more likely that multiple prescriptive standards converge (at least partially), since they use the same descriptive language. Thus, they only have to agree on terms, but not on the syntax used to define services and devices.

4 Optimality of Update Configurations

When smart home systems are updated, the objective of current systems is to install the latest software on all devices, regardless of dependencies or other objectives. While this is one possible approach, there exist others such as dependency robustness or flexibility of the smart home system. These different goals are much harder to achieve because additional constraints must be upheld. Single-objective optimization might be able to yield acceptable

results to some degree. Users might want to hold onto their existing automations, which in some cases could be damaged by the latest software policy.

If the goal is only to ensure that no functionality is broken, the solution is to simply pick the latest versions that do not remove any functionality currently in use. However, as soon as an update removes functionality, the problem is expanded to multiple objectives: It requires weighting between the benefits provided by the latest software version and the convenience of keeping all functionality unchanged. Determining an optimal solution becomes even more complicated when new functionality is introduced at the cost of another one. Zitzler et al. have compared evolutionary algorithms as a means to search for possible solutions while considering conflicting optimization goals [ZT99]. Solving multiple-objective optimization is a subject that has been explored for a long time, and a large number of possible solutions have been found [MA04; ZT99]. In this problem space the aim is not perfect optimization, which is commonly impossible, but for *Pareto optimality*. It is an approximation in which multiple configurations might be considered equal. This requires a choice between these solutions to be made, which can be implemented in the form of user choice between policies like *feature stability* and *security*. Equivalent results according to optimality can be presented to the user, who is then required to choose *a posteriori* [Br08].

However, the devices available in a smart home are usually also constrained in terms of performance or power. This can render such approaches unviable in the smart home context. To amend this situation, the user choice should be made *a priori* [MA04].

Self-description of services allows considering dependencies in a smart home analytically to find optimal update configurations for new objectives before the updates are installed. As it is possible to know which changes will happen once a certain update configuration is chosen, configurations that disrupt dependencies can be discarded. Static analysis over the available services and possible dependencies can be enhanced through a dynamic approach that tracks which services are actually used in a smart home. This way, services that are actually used can be considered during the update process, while services that are not used can be completely ignored in the search for an optimal configuration.

5 Autonomy of Smart Homes

User management, device communication, update planning, and automated tasks of a smart home might not occur locally but on remote servers. If the autonomy of a smart home is constrained by outsourcing functionality to external providers, the smart home depends on the availability of these providers. This availability cannot be guaranteed for the lifetime of smart homes. Despite this, it represents a common mode of operation for smart home systems currently in use, such as in the solutions of Amazon or Samsung.

External services (located in the cloud) can offer resources to solve computationally intensive tasks, manage authentication and other security-critical necessities, and offer a

gateway for remote access. These benefits provide a large incentive to waive autonomy in a smart home and assume that external services and internet connection are always available. An external service can even use approaches like testing updates of devices against their specification to reason about the correct functionality. It might also perform updates on specific configurations under laboratory settings. Nevertheless, if smart homes target lifetimes of at least 10 years and multiple vendors are involved, it becomes likely that some external services are shut down. Possible threats against the longevity of smart homes exist and have happened before [Zd20a; Zd20b].

5.1 Remote Update Planning

Giving away autonomy can be dangerous if update planning is performed remotely. To find an optimal update configuration, a smart home system must transmit information about all devices in the smart home, their installed software, and used dependencies to the remote service. Transferring information about usage habits in the form of dependencies and usage patterns is privacy-invasive. Specifics on the software installed on devices can disclose vulnerabilities currently open for exploitation at a location. By abusing the knowledge of vulnerable software on smart devices, access to various parts of the smart home could be gained and used to invade the privacy of users or even risk the security of the entire local network. Remote update planning can also pose additional dangers, as devices can be advised to update to a vulnerable software version, which might open up an attack vector. Waiving autonomy in a smart home must be considered carefully, as the impact depends on the task performed remotely.

5.2 Local Update Planning

Autonomy in a smart home requires local resources to solve problems that would otherwise be resolved with the help of a centralized external service. Smart home platforms like OpenHAB[7] strive to be autonomous, at the cost of much higher complexity. The higher complexity is a burden for non-technical users who feel overwhelmed by the amount of work necessary to configure and maintain a completely autonomous system.

To find the optimal update configuration autonomously, a suitable device must be available in the smart home. We will refer to this as the *central device*. This device needs to find out which other devices are part of the smart home, how they are connected, and which dependencies exist. Furthermore, the central device should monitor which functionality is actually being used in the smart home installation. Thus, the central device must query devices in the network or it must query local hubs, at least one for each platform in use. The practical problem of this approach is that some platforms provide no API to query this

[7] OpenHAB Documentation: `https://www.openhab.org/docs/` (accessed May 6, 2020)

information. While it is usually possible to enumerate all devices connected to a hub, it is often not possible to query which functionality is being used, or how these devices are connected among each other.

As smart home devices are created by numerous vendors using various platforms, a single database for available updates does not exist in general. The central device must therefore either query all device vendors for updates, or it must rely on the user to make updates available locally. Automatically querying device vendors implies that an external service is being used again. This implies that the smart home is leaking information about the devices deployed and the software versions installed to a wide range of device vendors. From a privacy perspective, this might be even more questionable than transmitting this information to a platform vendor like Apple, Amazon or Google, because these are at least known to the user, while users are usually not able to judge the trustworthiness of an oversea device manufacturer.

Once the central device has information about all possible updates including the self-description for all updates, it can compute which services are added or removed if a specific update is installed. Through optimality criteria, many of these update configurations can be discarded and the most advantageous configurations can be obtained. As stated before, multiple optimal configurations might exist. In the worst case, the number of possible options is $O(2^n)$ where n is the number of updates because this is the count of all possible subsets of updates. Obviously, it is not reasonable to present these options to the user.

Therefore, we propose a policy selection like *latest version*, *conservative*, or *feature set* to capture the intent of the user. This policy can be used to further filter the set of Pareto optimal configurations. Finally, the central device performs an update path that is Pareto optimal and this complies best to the chosen policy. Further research is required to actually develop and evaluate such a system to gain insight into the feasibility of this approach.

6 Conclusion

We analyzed the update problem for smart homes. The currently dominant approach is potentially dangerous, as not enough measures are taken to prevent harmful updates from being installed in a live system. Furthermore, the update process can potentially leak critical information, which violates privacy and can pose security risks for the entire network, because it might disclose attack vectors. We have shown that service definitions are required for the update planning and discussed descriptive and prescreptive approaches and their practicality. While autonomy is a desireable property for smart homes, local update planning is more complex than update planning that relies on cloud-based services of platform vendors. Finally, we discussed what optimality means for update planning and concluded that optimality alone is not sufficient to select an update path, since multiple optimal configurations can exist. We proposed policies to capture the user intent and to finally select one optimal update path.

Optimality criteria are not exclusive to smart homes. Dependency management of software projects can encounter similar issues. Hence, research in one of these domains might benefit both. As devices become more powerful, it might be possible to pull services running in the cloud into the home network and to deploy them in containers. Thus, smart homes could use external services for convenience, but lack no features if the cloud becomes unavailable or the user does not want to use external services.

References

[Br08] Branke, J.; Miettinen, K.; Deb, K.; Sowiski, R.: Multiobjective Optimization, Vol. 5252 of Lecture Notes in Computer Science. Multiobjective Optimization 5252/, pp. 1–8, 2008.

[Ka20] Kaebisch, S.; Kamiya, T.; McCool, M.; Charpenay, V.; Kovatsch, M.: Web of Things (WoT) Thing Description, first Edition of a Recommendation, https://www.w3.org/TR/wot-thing-description/, W3C, Apr. 2020.

[Ko20] Kovatsch, M.; Matsukura, R.; Lagally, M.; Kawaguchi, T.; Toumura, K.; Kajimoto, K.: Web of Things (WoT) Architecture, first Edition of a Recommendation, https://www.w3.org/TR/wot-architecture/, W3C, Apr. 2020.

[MA04] Marler, R. T.; Arora, J. S.: Survey of multi-objective optimization methods for engineering. Structural and multidisciplinary optimization 26/6, pp. 369–395, 2004.

[MR18] Mccool, M.; Reshetova, E.: Distributed Security Risks and Opportunities in the W3C Web of Things. In. Jan. 2018.

[Ve12] Vega-Barbas, M.; Casado-Mansilla, D.; Valero, M. A.; López-de-Ipiña, D.; Bravo, J.; Flórez, F.: Smart Spaces and Smart Objects Interoperability Architecture (S3OiA). In: 2012 Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing. Pp. 725–730, 2012.

[Zd20a] Zdankin, P.: Longevity of Smart Homes. In: PerCom PhD Forum 2020: 18th Annual IEEE International Conference on Pervasive Computing and Communications PhD Forum (PerCom PhD Forum 2020). Austin, USA, Mar. 2020.

[Zd20b] Zdankin, P.; Waltereit, M.; Matkovic, V.; Weis, T.: Towards Longevity of Smart Home Systems. In: PerIoT 2020: 4th International Workshop on Mobile and Pervasive Internet of Things (PerIoT 2020). Austin, USA, Mar. 2020.

[ZT99] Zitzler, E.; Thiele, L.: Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. IEEE Transactions on Evolutionary Computation 3/4, pp. 257–271, 1999.