Examining the Reusability of Smart Home Applications: A Case Study on Eclipse Smart Home

Paraskevi Smiari[1(✉)], Stamatia Bibi[1], and Daniel Feitosa[2]

[1] Department of Informatics and Telecommunications Engineering, University of Western Macedonia, Kozani, Greece
{psmiari, sbibi}@uowm.gr

[2] Data Research Centre, University of Groningen, Groningen, The Netherlands
d.feitosa@rug.nl

Abstract. Smart Homes consist of a plethora of IoT devices most of which developed by different manufacturers. To handle the diversity of IoT devices within the context of Smart Home automation, literature has suggested the use of frameworks. In this paper we argue that developers can benefit from such frameworks as a solution to build flexible and easily extendable systems by reusing their components. For this purpose, we explore the reuse opportunities that can be offered by Eclipse Smart Home (ESH) framework. In particular, we performed a case study and analyzed 107 packages from the ESH framework that offered 240 reusable components to the OpenHab application. We investigated (a) which types of functionality are mostly facilitated for reuse (b) which types of reuse are mostly adopted and what is the integration effort required (c) what is the quality of the reused components and compared them to the components built from scratch. The results of the case study suggest that: the main functionality reused is the one related to Interface Adapters and the main type of reuse is Variable Type. Regarding the effort for integrating the reused components it can range from 38 lines of code to 1421 lines of code. Moreover, the quality of the reused components is slightly improved compared to the rest of the components built from scratch.

Keywords: Smart Home · IoT applications · Reusability · Effort estimation · Flexibility · Extendibility

1 Introduction

Since the last decade, the emerging paradigm of the Internet of Things (IoT) dominates the Smart City [15] landscape, offering a range of citizen services [34] that include among others, healthcare, education, energy consumption, and home automation. According to ABI[1], the research of the realization of the Smart City concept should start from the Smart Home. At the moment the Smart Home revenue amounts at 69,551

[1] https://www.smartcitiesworld.net/special-reports/special-reports/why-the-smart-city-could-increasingly-start-at-home.

X. Peng et al. (Eds.): ICSR 2019, LNCS 11602, pp. 232–247, 2019.
https://doi.org/10.1007/978-3-030-22888-0_16

million dollars and is expected to present an annual growth of 20.3%, while the penetration of Smart Home technologies in households is still in its infancy. By 2019, 9.5% of households have already adopted Smart Home technologies, a percentage that is expected to hit 22.1% by 2023[2]. IoT devices are the key concept in Smart Homes to manage energy, security, lighting and appliances [1]. Smart Homes consist of a plethora of IoT devices (i.e., sensors and meters) that need to cooperate and are mostly developed by different manufacturers, as well as designed and implemented with heterogeneous technologies. Therefore, Smart Home applications need to be flexible so as to manipulate the individuality of each device [12], and also extendable so as to integrate new devices [28].

To handle this inherit diversity of IoT devices within the context of Smart Home automation, literature has suggested the use of frameworks that are based on highly modularized software building blocks [11, 12, 25]. In these frameworks, despite the fact that they are very large and complex, one can identify a core set of concepts (i.e. Devices and Controllers) which can enable reuse. Therefore, developers can benefit from such frameworks as a solution to build flexible and easily extendable systems by reusing their components. Currently there are several available Open Source Software frameworks[3] with big support from the community that can facilitate reuse. The main challenges that an engineer confronts while reusing components from one of these frameworks are summarized as follows:

(a) Select the ***type of functionality to reuse.*** The functionalities offered by Smart Home frameworks can be classified based on the requirement that they implement. For example, a component can be related to a core Smart Home purpose (i.e., a Device) and therefore increase its potential reuse or can implement specific details (i.e., Visualization Graphs) and decrease its potential reuse.
(b) Plan the ***type of reuse*** and the ***effort*** required for integrating the reused components. The ***type of reuse*** can vary from simply using components as they are, to implementing new functionalities on the reused components for integrating them. In the last case it is important to have an approximation of the ***effort*** required to integrate the reused component, which can be measured as the time required to apply the changes or the amount of new lines of code added.
(c) Ensure that the ***quality of the reusable components*** does not compromise the overall quality of the application. For this purpose, it is important to examine the quality aspects that are important for Smart Home applications and make sure that they do not present quality differences with the rest of the application developed from scratch.

In this study we explore how the aforementioned challenges are confronted when reusing components from the popular Eclipse Smart Home (ESH) framework that is used as the 'source' application. ESH was selected because it is frequently used in research [19, 21, 33], has a strong support from industrial players like Bosch and QIVICON, and is often reused for building commercial products (e.g., Mixtile Hub,

[2] https://www.statista.com/outlook/279/100/smart-home/worldwide.

[3] https://www.eclipse.org/smarthome/, https://www.openhab.org/, https://www.home-assistant.io/.

Coqon). As a 'target' application, we employ the OpenHab project, which is based on ESH. OpenHab takes advantage of the abstraction level that ESH provides and offers a holistic platform that supports different devices under one design. As part of this study we analyzed 107 packages from the ESH framework, which offers 240 reusable components to OpenHab. Our results showed that the main functionality reused is the one related to Interface Adapters, the main type of reuse is Variable Type and the effort for integrating the reused components can range from 38 lines of code to 1421 lines of code. Moreover, the quality of the reused components is slightly improved in comparison to the rest of the components built from scratch; however, without presenting a statistically significant difference.

The rest of this paper is organized as follows. Section 2 discusses the related work. Section 3 describes the case study design whereas Sect. 4 presents the results obtained from the case study. Finally, Sect. 5 discusses the results and addresses the threats to validity of this study and Sect. 6 concludes the paper and presents ideas for future work.

2 Related Work

Software reuse is a widely known and used technique for the creation of a new software product that is based on the adoption of existing software components [30]. Among the major benefits offered by software reuse is the minimization of the cost and the effort required to develop an application [13]. Several studies [26, 31] mention that increased productivity along with cost reduction is among the main objectives of software reuse. This is achieved through placing emphasis on identifying and integrating reusable components instead of designing and implementing new functionalities from scratch [6, 16]. Another benefit of software reuse is the improvement of software system flexibility [10, 31]. According to Jatain et al., [10] development based on components increases significantly the system flexibility. This is achieved through the separation of the stable parts of systems from the specification of their composition [10]. Software reuse is also related to the increased quality of the final product [26] in the sense that the reuse of software components that are already tested and validated, increases the probability of developing applications of increased quality [26].

Discovering the reusable software is a challenge itself as it is important to efficiently prioritize candidate assets for reuse and select the most appropriate ones [2]. Research has focused on data mining techniques [22, 32] for identifying reusable assets by taking into consideration several software quality metrics. Evaluating reusable components through a set of metrics and predicting whether or not are qualified as reusable [17] is very important when it comes to discovering reusable software. Moreover, clustering techniques for identifying components of similar functionality and reusability have been thoroughly discussed by Jatain et al. in [10].

Another major challenge that arises when it comes to the reusability of the software is adapting the components that are being reused into the new environment that is being developed [27]. It is often required to parameterize the reusable components in order to integrate them in the existing environment [9, 18]. Refactoring in a component that is being reused, or in the project that reuses it, can have the opposite effect and can be

time consuming, especially if the new environment is demanding [5]. Therefore, it is important to place special emphasis on carefully planning the integration of the reusable artifact into the new application [7].

To the best of our knowledge, the practice of software reuse in the IoT development process has not been explored so far, despite the numerous benefits that it offers. In this study we go beyond current literature by examining the software reuse potentials for building home automation IoT applications. In particular, we investigate to which level the aforementioned reuse goals are achieved and explore the challenges of reuse that are related to the quality of the reused components and the effort required to integrate the components to the new environment.

3 Case Study Design and Evaluation

In this section, we present the design of the case study performed to assess the reusability potentials of IoT frameworks. We report the details of this case study based on the guidelines of Runeson et al. [24]. In Sect. 3.1, we present the research objectives of the study. In Sects. 3.2 and 3.3, we describe the two applications that participate in this study and the data collection processes. Finally, we provide an overview of the data analysis process in Sect. 3.4.

3.1 Research Objectives and Questions

The overall goal of this case study is to ***examine the reuse potentials for IoT home automation application frameworks*** in terms of (a) the ***type of functionality*** of the component that is being reused (b) the level of ***customization*** required for integrating the reused component and (c) the ***quality*** of the final product. To ease the design and reporting of the case study, we split the aforementioned goal into five research questions, based on the analysis perspectives that we introduced in Sect. 1, as follows:

*RQ1: Which **types of functionality** offer the most components in the context of IoT **home automation** application reuse?*

This research question aims at identifying the types of functionalities that offer the larger pool of components. The classification scheme adopted to assess the types of the functionalities offered by the reused components was inspired from the concept of "clean architecture" [14]. The discipline of Clean Architecture defines four core types of functionalities namely, the Enterprise Business Rules, the Application Business Rules, the Interface Adapters, and the Frameworks & Drivers [14]. The analysis in this question will provide an overall view of the number of the reused components offered by each type of functionality.

*RQ2: Which **types of reuse** offer the most components in the context of IoT **home automation** application reuse?*

This research question aims at identifying the *types of reuse* that are mostly adopted in the context of IoT applications. We consider three types of reuse, Variable type reuse, Static method or Constant value reuse, and Implementation of interface or parent class

(see Sect. 3.3). The analysis in this question will provide an overall view of which types of reuse is expected to be adopted.

RQ3: What is the level of customization required to integrate the reused components?

This research question explores the effort required to integrate a reused component in the new project. The integration effort, in our case, is measured as the lines of code for implementing new functionalities related to the reused artifacts (see analytically in Sect. 4). The output of this research question will provide insights on the level of changes required for each type of functionality reused.

RQ4: What is the quality of the reused components with respect to the functionalities offered?

This research question investigates the quality of the reused components by assessing four indices of the QMOOD model [3], *Reusability*, *Flexibility*, *Extendibility* and *Functionality*. We selected these quality aspects as two of them (Reusability and Functionality) are highly related to the reusability of the new application [4, 20], while the other two (Flexibility and Extendibility) are appointed as important quality aspects of IoT applications [28]. The result of this research question will be an analysis of the quality of the reused components per type of functionality offered.

RQ5: Is there a difference between the quality of the components developed with reuse and the quality of the components developed from scratch and how is this reflected to the different types of the functionalities offered?

This research question compares the quality of the components developed with reuse and the quality of the components developed from scratch, by assessing the four indices of the QMOOD model [3]: *Reusability*, *Flexibility*, *Extendibility* and *Functionality*. The result of this research question will provide insights on whether reuse of components in IoT application development is expected to bring quality benefits. As a second step, we will diversify between the quality of components built with and without reuse with respect to the offered functionalities.

3.2 Case Selection

This section presents the details of the two projects that were selected for examining the reuse potentials in IoT applications. The first part of this section presents the context of the Eclipse Smart Home application, which is considered as the 'source' of the reusable components. The second part of this section presents the context of OpenHab 2, which is considered as the 'target' of the reusable components.

Eclipse Smart Home

The **Eclipse Smart Home (ESH)** project is an open source framework for developing home automation IoT applications[4] that have a strong focus on heterogeneous environments. It is comprised of a set of OSGi bundles that support the integration of different protocols and standards. ESH was launched in 2014 and it is part of the

[4] https://www.eclipse.org/smarthome/, https://iot.eclipse.org/.

Eclipse IoT, which comprises four projects providing open source implementations of IoT protocols, services and frameworks. The community of ESH is composed of over 179 active contributors, while more than six companies have adopted this framework for developing on top IoT applications. Currently, the project has been forked over 829 times and has a history of four releases.

In this study, we explored ESH, version ref-0.10.0 as the 'source' that provides the reusable components for building IoT applications on top. Currently, ESH provides all the benefits of an open source project, such as a big support from the community and cost efficiency which renders it a primary choice when it comes to reusing components for building IoT solutions. Additionally, ESH is consistently used for empirical research the last years in the context of building automations [19], monitoring environments for smart and secure homes [21, 33], and from the perspective of user interface [29]. Table 1 presents the summary statistics of the two projects adopted in this study.

OpenHab2

The **Op**en **H**ome **A**utomation **B**us (OpenHab) is an open source, technology agnostic home automation platform launched on February 2010. The platform has been actively maintained in the last 9 years and it is a solution built upon the ESH framework. The founders of OpenHab, afterwards, donated the core framework of the application to the Eclipse Foundation so as to benefit from the rigid intellectual property management and clear contribution processes that Eclipse provides. For the purposes of this study, we use the latest version of OpenHab 2[5] (v2.4). The functionalities offered by OpenHab 2 are split into two projects: OpenHab Core and OpenHab AddOns. The OpenHab Core is the repository in which all the core framework bundles are implemented, whereas OpenHab AddOns is a repository that contains all the bindings and services. Bindings are responsible for integrating physical hardware, external systems, and web services in OpenHab.

Table 1. Eclipse Smart Home and OpenHab projects.

Project	Packages	Nof. classes	LoC	Releases
Eclipse S. Home	107	2197	363.735	4
Eclipse S. Home (r)*	47	240	139.174	4
OpenHab	147	3384	394.788	5
OpenHab (wr)**	140	704	259.661	5

* Statistics of the Eclipse S. Home packages, classes, etc. being reused

** Statistics of the OpenHab packages, classes, built with reuse

[5] https://www.openhab.org/.

In this case study, OpenHab is the 'target' of the reused components and will serve as a sample to explore the benefits acquired from reuse with respect to the different types of functionalities reused. We selected to explore the level of reuse performed in the context of the development of OpenHab because: (a) it is a fully functional and 'open' platform that can be considered as an end-product instead of a framework (that is the case of Eclipse Smart Home); (b) it is open source, which allows as to fully explore all the types of reuse potentials; (c) it is very popular, counting almost 30,000 members, supporting the interface of over 1,500 IoT devices and gadgets; and (d) it can be considered as a representative of closed-source commercial projects (e.g., Mixtile, Qivicon) that are based on Eclipse Smart Home Framework to implement end-product solutions.

3.3 Data Collection

In order to answer the research questions of this study we followed the process summarized in the next steps:

Step1: We calculated the ***Actual Reuse*** of each component of ESH, using the information reported in the Maven[6] repository. We isolated the packages that are being reused, mapped these packages to the related OpenHab packages and counted the *Number of Components* that each ESH package offers and the *Lines of Code* offered to OpenHab.

Step2: The second step was to classify the ***Type of Functionality*** offered from ESH reused packages into the four categories of Clean Architecture, Enterprise Business Rules, Application Business Rules, Interface Adapters, and Frameworks & Drivers. For this purpose, we extracted the semantics behind every ESH class. For example, the class Thing represents everything that can be physically added to the system. It is one of the core classes in ESH and part of the core architecture which makes it an Entity and, therefore, part of the Enterprise Business Rules. In Table 2 we provide the keywords used to classify each class into one of the four categories of functionalities.

Step3: As a next step we classified the ***Type of Reuse*** in one of the following classes: Variable type reuse, Static method or Constant value reuse, and Implementation of interface or parent class. By *Variable type* reuse we mean an instantiation of a new object from the reused class that has not been subject to changes. *Static method or constant value reuse* refers to the utilization of a parameter or function of a reused class that has not been subject to changes. *Implementation of interface, or parent class* refers to the extension of the reused class by adding new functionalities or implementing existing definitions.

Step 4: Next, we recorded the ***New Lines of Code*** developed for integrating the reused components. For this purpose, we counted the lines of code developed in the case of *Implementation of interface, or parent class* type of reuse, considering that in the other two types of reuse no changes have been performed, since in the cases

[6] https://mvnrepository.com/.

were OpenHab adds new functionalities in an ESH class always occurs from the inheritance of a class.

Step 5: As a final step we recorded the ***Quality*** of the Components participating in the reuse process by calculating the metrics defined by Bansiya et al. for assessing the ***Functionality***, the ***Extendibility***, the ***Reusability,*** and the ***Flexibility*** [3]. We calculated these four quality metrics for the Reused (R) components coming from ESH and for the components developed With Reuse in the OpenHab project. To have a holistic view of the quality obtained with and without reuse, we also calculated the Quality of the Components developed Without Reuse (WR). Table 2 presents the metrics considered within the scope of this study and their description.

Table 2. Design metrics

Type	Metric	Description
Reuse metrics	Development with reuse	Shows if a component has been built with reuse (Yes/No)
	Type of functionality	Enterprise Business Rules (keyword = Thing) Application Business Rules (keyword = Configuration) Interface Adapters (keyword = Handler) Frameworks & Drivers (keyword = SiteMap)
	Type of reuse	Variable Type Static method call or Constant value Implementation of Interface or Parent class
	Number of components (NoC)	Number of components reused (each component corresponds to one class)
	Total reuse of components (TRC)	Total number of times a component is reused in OpenHab
Effort metrics	New lines of code	Lines of code added by OpenHab
	Lines of code reused	Lines of code provided by reusable components of ESH
Quality metrics	Functionality	0.12 * CAM + 0.22 * NOP + 0.22 * CIS + 0.22 * DSC + 0.22 * NOH
	Extendibility	0.5 * ANA-0.5 * DCC + 0.5 * MFA + 0.5 * NOP
	Reusability	−0.25 * DCC + 0.25*CAM +0.5 * CIS + 0.5 * DSC
	Flexibility	0.25 * DAM-0.25 * DCC + 0.5 * MOA + 0.5 * NOP

CAM = Cohesion Among Methods of Class, NOP = Number of Polymorphic Methods, CIS = Class Interface Size, DSC = Design Size in Classes, NOH = Number of Hierarchies, ANA = Average Number of Ancestors, DCC = Direct Class Coupling, MFA = Measure of Functional Abstraction, DAM = Data Access Metric, MOA = Measure of Aggregation [3]

3.4 Data Analysis

The data analysis of this case study includes the calculation of the frequency and descriptive statistics, and the application of Analysis of Variance (ANOVA).

For ***RQ1,*** we provide the frequency statistics of the total number of components and packages offered by ESH with respect to their *Type of Functionality*. Additionally, ANOVA is performed to identify whether the *Number of Components* offered by the different *Types of Functionality* varies significantly.

Concerning ***RQ2,*** we discuss the frequency with which the different *Types of Reuse* are implemented by providing the relevant pie chart.

For addressing RQ_3 the descriptive statistics (min, max, mean, st.dev) are presented for the offered *Type of Functionality* and the *Lines of Code* required for integrating the reused components. In this case, ANOVA is performed to identify whether different types of functionality offer components that present *significant* differences in the effort required for their integration.

In ***RQ4,*** we perform ANOVA to identify whether there are significant differences in the quality *(Extendibility, Flexibility, Reusability, Functionality*) of the provided components for the different *Types of Functionality*. The descriptive statistics are also presented.

Similarly, in ***RQ5,*** we perform ANOVA to identify whether there are significant differences in the quality *(Extendibility, Flexibility, Reusability, Functionality*) of the packages developed with reuse and the packages developed without reuse. In this case, the grouping variable is the *Development with Reuse.*

4 Results

In this section, we present and interpret the results of this case study, organized by research question, and based on the data analysis presented in Sect. 3.4.

RQ1 – Which types of functionality offer the most components in the context of IoT home automation application reuse?

Table 3 presents the summary statistics for the four types of functionality offered by ESH, ranked by the frequency of the reused components (see RF - column 5). It can be observed that the highest number of components (see CR – column 4) offer functionality related to Frameworks & Drivers and Interface Adapters. In terms of the highest reuse frequency per functionality type components (see RF – column 5), we observed that the maximum percentage is recorded for two functionality types: Application Business Rules and Frameworks & Drivers. On the other hand, the least reused components offer functionality related to Enterprise Business Rules.

Table 3. Reuse per type of functionality

Type of functionality	TP	TC	PR	CR	RF
Application Business Rules	22	214	16	53	0.25
Frameworks & Drivers	42	567	26	97	0.17
Interface Adapters	38	1224	25	81	0.07
Enterprise Business Rules	5	192	1	9	0.05

TP: Total number of packages in ESH per functionality type.
TC: Total number of components per functionality type.
PR: Number of packages of ESH reused by OpenHab per functionality type.
CR: Number of components of ESH reused by OpenHab per functionality type.
RF: Reuse Frequency = CR/TC

To investigate if the aforementioned differences are statistically significant, we performed an Analysis of Variance (ANOVA), which suggested that there are not significant differences in the reusable components of the different types of functionality (F: 2.276, sig: 0.087).

RQ2 – Which types of reuse offer the most components in the context of IoT home automation application reuse?

Concerning RQ2, we discuss the frequency of the different Types of Reuse (see Fig. 1). The results of the pie chart suggested that most of the reused components are used directly as Variable types or Calls to static methods/constant values. However, the implementation of new functionality was required in 24.2% of the reused components. It seems that in the majority of the cases reuse is performed without any changes of integration effort. This can be interpreted intuitively since, as mentioned in RQ1, the majority of the reused components offer core functionality (i.e., Application Business rules and Enterprise Business Rules), which are the least likely to require changes.

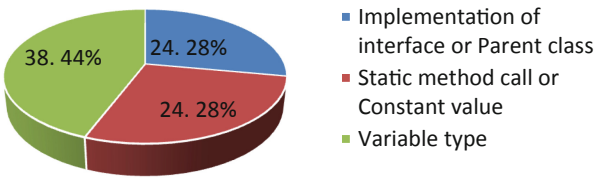

Fig. 1. Pie Chart (Frequency of usage types)

RQ3 – What is the level of customization required to integrate the reused components?

To investigate the effort required for the integration of the components with respect to the different types of reused functionality, we calculated the descriptive statistics and performed ANOVA for the variable *New Lines of Code,* but only for the cases where the reuse type is Implementation of interface/parent class. In Table 4 we present the summary statistics of the effort required to integrate the components of different functionality types. The first column shows the value of ***TRC*** variable that indicates accumulatively the total number of the times that the components of different functionality types have been reused. It can be observed in Table 4 that the type of functionality that requires the greatest effort on average is Interface Adapters. This result can be interpreted intuitively since Interface and Adapters components are responsible for integrating and supporting a wide range of smart devices, a fact that requires certain code for the customization of the generic components being reused.

Table 4. Effort required to integrate the reused components per type of functionality.

Type of functionality	TRC	Mean	Std. Dev	Min	Max
Enterprise Business Rules	1	72.00	–	72	72
Application Business Rules	9	208.44	95.06	94	387
Interface Adapters	357	237.37	221.29	42	1421
Frameworks & Drivers	144	155.23	85.02	38	662

To investigate if the aforementioned differences are statistically significant, we performed an ANOVA (F: 10.140, Sig: < 0.001), which showed a significant difference between groups in OpenHab with respect to the effort required when it comes to their integration.

RQ4 – What is the quality of the reused components with respect to the functionalities offered?

In Table 5 we present the results of the quality of the reused components provided by ESH.

Table 5. Quality of the ESH and OpenHab components per type of functionality offered.

Type of functionality	Quality	ESH	OpenHab
Enterprise Business Rules	Reusability	15.71	6.22
	Flexibility	0.07	0.25
	Functionality	5.67	3.02
	Extendibility	0.60	0.1
Application Business Rules	Reusability	10.19	90.73
	Flexibility	0.15	1.00
	Functionality	3.28	42.06
	Extendibility	0.32	1.51
Interface Adapters	Reusability	12.16	12.28
	Flexibility	0.37	0.26
	Functionality	3.43	6.19
	Extendibility	0.30	0.50
Frameworks & Drivers	Reusability	9.65	11.67
	Flexibility	0.13	0.28
	Functionality	2.93	5.57
	Extendibility	0.09	0.34

For this research question, we examined *Reusability, Flexibility, Functionality, and Extendibility* with respect to the offered functionalities. In terms of *Reusability,* the differences are small, with the exception of Enterprise Business Rules, which is observed to have a higher mean value in comparison to the other functionality types. Additionally, the results of ANOVA calculated between groups (F: 0.196, Sig: 0.899)

confirmed that there is no significant difference between groups. In terms of *Flexibility,* the differences are also small, with the exception of Interface Adapters having higher mean value in *Flexibility* than the other types of functionality. After the calculation of ANOVA, the results (F: 0.532, Sig: 0.663) do not suggest the existence of significant differences. In terms of *Functionality,* we observed one noticeable distinction about Enterprise Business Rules, having a somewhat higher mean value, although only one package can be described with that functionality. The results of ANOVA (F: 0.345, Sig: 0.793) did not suggest the existence of significant differences. Concluding with *Extendibility*, the differences are negligible, and Enterprise Business Rules display the highest mean value, while the results of ANOVA calculated between groups (F: 1.953, Sig: 0.135) did not suggest the existence of significant differences.

In Table 5 we also present the results of the quality of the components developed with reuse in OpenHab. In terms of *Reusability* we observed difference in the mean values with higher being the Application Business Rules. Additionally, the results of ANOVA calculated between groups (F: 2.620, Sig: 0.05) suggested the existence of significant differences. In terms of *Flexibility* the difference between the mean values across packages offering different functionality types is very small, a fact that is also confirmed by the calculation of ANOVA (F: 0.155, Sig: 0.927). Furthermore, in terms of *Functionality* the results show that there is no significant difference, ANOVA (F: 2.285, Sig: 0.082), between functionality types with Application Business Rules having the highest mean value. Concluding with *Extendibility* we can observe that Application Business Rules and Interface Adapters have the highest mean values though there isn't a statistically significant difference between functionality types ANOVA (F: 1.428, Sig: 0.237).

RQ5 – Is there a difference between the quality of the components developed with reuse and the quality of the components developed from scratch and how is this reflected to the different types of the functionalities offered?

In Table 6 we compared the quality of the OpenHab packages that were developed with reuse (140 packages) to those developed without reuse (7 packages). In terms of *Reusability* there is a difference between components developed with reuse and components developed from scratch. We observed that components developed with reuse have a significantly higher value, as confirmed by the calculation of ANOVA (F: 6.096, Sig: 0.015). In terms of *Flexibility* the packages developed with reuse showed higher values. From the calculation of ANOVA (F: 5.224, Sig: 0.024), we noticed a significant difference between groups. In terms of *Functionality,* we observed a substantial difference, in which the components developed with reuse show a greater average value with the ANOVA (F: 6.194, Sig: 0.014) confirming the significant difference. The final quality metric we explored is Extendibility, which did not show differences between the two groups, although components developed from scratch showed a higher value. After the calculation of ANOVA (F: 0.419, Sig: 0.519).

Table 6. OpenHab per quality type

Developed with reuse	Reusability	Flexibility	Functionality	Extendibility
No	2.00	0.14	0.89	0.57
Yes	15.74	0.26	7.69	0.48

5 Discussion

The results of this paper revealed that the top two types of component functionality that are more likely to be reused in the context of Smart Home application development are: *Application Business Rules and Interface Adapters*. *Application Business Rules* and *Enterprise Business Rules* types of functionality are the least likely to require integration effort as they are usually implemented as Variable type or Static method call/constant value. On the other hand, components implementing Interface Adapters are the ones that require significantly more effort to be integrated in the new application (on average, 234 Lines of Code). Regarding the quality of the reused components, it is observed that there are no significant differences between the components offering different types of functionality. Finally, regarding the difference between the quality of the packages build with reuse and the packages built without reuse we observe that there are significant differences in terms of *Functionality, Reusability and Flexibility,* showing that software reuse can lead to increased quality of the application.

The results of this study provide useful information and guidance to ***practitioners*** on planning the reuse of components in the context of Smart Home application development. In particular, some general conclusions that we reached from this case study are:

- Engineers of IoT, Smart Home applications can greatly benefit from reusing a core set of general purpose components, which in our case are the *Enterprise Business Rules* and the *Applications Business Rules*. These components, in their majority, can be reused as is without requiring any integration effort.
- The reuse of components offering functionality related to *Interface Adapters* can also be beneficial, since this type of functionality offered the most reused components. However, reusing components related to Interface and Adapters required certain integration efforts for implementing or extending the classes reused.
- In terms of the four examined quality characteristics, components of type Interface Adapters seem to present the highest potentials of being reused. This can be interpreted intuitively, as these types of components should be hardware-agnostic and abstract the details of the Application Business Rules.

Based on the results of this case study, we encourage ***researchers*** to:

- Further explore the reuse of components in the context of IoT application development for Smart Home automation by examining other Open Source projects (e.g., Home Assistant). Researchers can investigate whether the same type of components, as appointed by this study, have been systematically reused. The results of

this study can also guide researchers in assessing the appropriateness of the reused components.
- Introduce a process for systematic, planned reuse of IoT components. Such a process would define clear procedures for: (a) identifying and sorting the reusable components, (b) integrating the reused components into the new applications, and (c) maintaining these components.

To conclude this section, we refer to the ***threats to validity*** of this case study [24]. A possible threat to ***construct validity*** is related to the metrics that are used to answer our research questions. Regarding the effort metrics, we believe that the new lines of code are indicators of the effort required to integrate the reused components. This metric has been also adopted in [8] and [23] for assessing the reuse effort. Nevertheless, we acknowledge that there are other metrics that can also be used. For the quality assessment of the reused components, we have used QMOOD, which is an established quality model that has been rigorously validated [3]. However, we acknowledge that other quality models could lead to variations in the observed results. With regard to ***reliability***, we acknowledge potential researchers' bias during the data collection due to the manual classification of components into types of functionality performed by the first author. To mitigate reliability threats the second and third author validated the results. Finally, we acknowledge that the ***external validity***, is threatened by the fact that the entire data set is taken from one single reuse case between Eclipse Smart Home and OpenHab. However, we believe that the results can be generalize in the context of reuse of Smart Home automation frameworks since the majority of applications are built from modular blocks that can be easily classified in the functionality types employed in this study [14]. Threats to ***internal validity*** are not discussed in this paper, as we did not seek to identify causal relations in this study.

6 Conclusions

In this paper, we explored the reuse opportunities stemming from the popular Eclipse Smart Home framework for building home automation IoT applications. We performed a case study and investigated the types of functionality that can be reused, the effort required for integrating the reused components and whether or not such reuse leads to quality benefits. We analyzed 107 packages from the ESH framework and 240 reused components from the OpenHab application. The results of this case study suggest that: the main reused functionality is related to Interface Adapters; the main type of reuse is Variable Type; and the effort for integrating the reused components can range from 38 lines of code to 1421 lines of code. The quality of the reused components is slightly higher compared to components built from scratch. As future work we intend to further explore reuse opportunities within home automation IoT frameworks by examining other open source frameworks (e.g., Home Assistant), retrieving candidate components, and comparing them.

Acknowledgement. This research was co-funded by the European Union and Greek national funds through the Operational Program Competitiveness, Entrepreneurship, and Innovation, grant number T1EDK-04873.

References

1. Alam, M.R., Reaz, M.B.I., Ali, M.A.M.: A review of smart homes—past, present, and future. IEEE Trans. Syst. Man Cybern. Part C (Appl. Rev.) **42**(6), 1190–1203 (2012)
2. Ayala, C., Hauge, Ø., Conradi, R., Franch, X., Li, J.: Selection of third party software in Off-The-Shelf-based software development—an interview study with industrial practitioners. J. Syst. Softw. **84**(4), 620–637 (2011)
3. Bansiya, J., Davis, C.G.: A hierarchical model for object-oriented design quality assessment. IEEE Trans. Softw. Eng. **28**(1), 4–17 (2002)
4. Benni, B., Mosser, S., Moha, N., Riveill, M.: A delta-oriented approach to support the safe reuse of black-box code rewriters. In: Capilla, R., Gallina, B., Cetina, C. (eds.) ICSR 2018. LNCS, vol. 10826, pp. 164–180. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-90421-4_11
5. Brereton, P., Budgen, D.: Component-based systems: a classification of issues. Computer **33** (11), 54–62 (2000)
6. Caldiera, G., Basili, V.R.: Identifying and qualifying reusable software components. Computer **24**(2), 61–70 (1991)
7. Crnkovic, I., Larsson, M.: Challenges of component-based development. J. Syst. Softw. **61** (3), 201–212 (2002)
8. Gui, G., Scott, P.D.: Coupling and cohesion measures for evaluation of component reusability. In: Proceedings of the 2006 International Workshop on Mining Software Repositories, pp. 18–21. ACM (2006)
9. Gupta, A., Cruzes, D., Shull, F., Conradi, R., Rønneberg, H., Landre, E.: An examination of change profiles in reusable and non-reusable software systems. J. Softw. Maint. Evol. Res. Pract. **22**(5), 359–380 (2010)
10. Jatain, A., Nagpal, A., Gaur, D.: Agglomerative hierarchical approach for clustering components of similar reusability. Int. J. Comput. Appl. **68**(2), 33–37 (2013)
11. Kamilaris, A., Trifa, V., Pitsillides, A.: HomeWeb: an application framework for Web-based smart homes. In: 2011 18th International Conference on IEEE Telecommunications (ICT), pp. 134–139 (2011)
12. Kim, J.E., Boulos, G., Yackovich, J., Barth, T., Beckel, C., Mosse, D.: Seamless integration of heterogeneous devices and access control in smart homes. In: 2012 8th International Conference on IEEE Intelligent Environments (IE), pp. 206–213 (2012)
13. Ma, S., Yang, H., Shi, M.: Developing a creative travel management system based on software reuse and abstraction techniques. In: 2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC), vol. 2, pp. 419–424. IEEE (2017)
14. Martin, R.C.: Clean Architecture: A Craftsman's Guide to Software Structure and Design. Prentice Hall Press, Upper Saddle River (2017)
15. Mehmood, Y., Ahmad, F., Yaqoob, I., Adnane, A., Imran, M., Guizani, S.: Internet-of-things-based smart cities: recent advances and challenges. IEEE Commun. Mag. **55**(9), 16–24 (2017)
16. Ostertag, E., Hendler, J., Prieto-Díaz, R., Braun, C.: Computing similarity in a reuse library system: an AI-based approach. ACM Trans. Softw. Eng. Methodol. **1**(3), 205–228 (1992)

17. Padhy, N., Singh, R.P., Satapathy, S.C.: Software reusability metrics estimation: algorithms, models and optimization techniques. Comput. Electr. Eng. **69**, 653–668 (2018)
18. Pacheco, C.L., Garcia, I.A., Calvo-Manzano, J.A., Arcilla, M.: A proposed model for reuse of software requirements in requirements catalog. J. Softw. Evol. Process **27**(1), 1–21 (2015)
19. Panwar, A., Singh, A., Kumawat, R., Jaidka, S., Garg, K. Eyrie smart home automation using Internet of Things. In: 2017 Computing Conference, pp. 1368–1370. IEEE (2017)
20. Paschali, M.-E., Ampatzoglou, A., Bibi, S., Chatzigeorgiou, A., Stamelos, I.: A case study on the availability of open-source components for game development. In: Kapitsaki, G.M., Santana de Almeida, E. (eds.) ICSR 2016. LNCS, vol. 9679, pp. 149–164. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-35122-3_11
21. Perera, C., McCormick, C., Bandara, A.K., Price, B.A., Nuseibeh, B.: Privacy-by-design framework for assessing internet of things applications and platforms. In: Proceedings of the 6th International Conference on the Internet of Things, pp. 83–92. ACM (2016)
22. Prakash, B.A., Ashoka, D.V., Aradhya, V.M.: Application of data mining techniques for software reuse process. Procedia Technol. **4**, 384–389 (2012)
23. Prieto-Diaz, R., Freeman, P.: Classifying software for reusability. IEEE Softw. **4**(1), 6 (1987)
24. Runeson, P., Höst, M.: Guidelines for conducting and reporting case study research in software engineering. Empirical Softw. Eng. **14**(2), 131 (2009)
25. Serna, M.A., Sreenan, C.J., Fedor, S.: A visual programming framework for wireless sensor networks in smart home applications. In: 2015 IEEE Tenth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), pp. 1–6 (2015)
26. Sharma, A., Grover, P.S., Kumar, R.: Reusability assessment for software components. ACM SIGSOFT Softw. Eng. Notes **34**(2), 1–6 (2009)
27. Singh, S., Singh, S., Singh, G.: Reusability of the software. Int. J. Comput. Appl. **7**(14), 38–41 (2010)
28. Smiari, P., Bibi, S.: A smart city application modeling framework: a case study on re-engineering a smart retail platform. In: 2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), pp. 111–118. IEEE (2018)
29. Smirek, L., Zimmermann, G., Beigl, M.: Just a smart home or your smart home–a framework for personalized user interfaces based on eclipse smart home and universal remote console. Procedia Comput. Sci. **98**, 107–116 (2016)
30. Vale, T., Crnkovic, I., De Almeida, E.S., Neto, P.A.D.M.S., Cavalcanti, Y.C., de Lemos Meira, S.R.: Twenty-eight years of component-based software engineering. J. Syst. Softw. **111**, 128–148 (2016)
31. Varadan, R., Channabasavaiah, K., Simpson, S., Holley, K., Allam, A.: Increasing business flexibility and SOA adoption through effective SOA governance. IBM Syst. J. **47**(3), 473–488 (2008)
32. Wangoo, D.P., Singh, A.: A classification based predictive cost model for measuring reusability level of open source software (2018)
33. Wen, X., Wang, Y.: Design of smart home environment monitoring system based on raspberry Pi. In: 2018 Chinese Control and Decision Conference (CCDC), pp. 4259–4263. IEEE (2018)
34. Zanella, A., Bui, N., Castellani, A., Vangelista, L., Zorzi, M.: Internet of things for smart cities. IEEE Internet Things J. **1**(1), 22–32 (2014)