ISSN 2319 – 7595
Volume 8, No.6, November - December 2019
International Journal of Information Systems and Computer Sciences
Available Online at http://warse.org/IJISCS/static/pdf/file/ijiscs01862019.pdf
https://doi.org/10.30534/ijiscs/2019/01862019

Eliciting Security Requirements of IoT Applications using Essential Use Case

Asma Asdayana Ibrahim[1], Massila Kamalrudin[2]
[1] Faculty of Information and Communication Technology,
Universiti Teknikal Malaysia Melaka, 76100, Malaysia,
asmaasdayana@gmail.com
[2] Innovative Software System and Service Group,
Universiti Teknikal Malaysia Melaka, 76100, Malaysia,
massila@utem.edu.my

ABSTRACT

Requirement elicitation is important to ensure correct requirements are elicit for application development. Wrong elicitation decision leads to a software failure. The most common challenges task is to ensure consistent requirements are elicit between engineers and the users. Most requirements engineers face difficulties to fulfill the consistency with clients especially in eliciting IoT security requirements as it involves many components such as devices, domains and security attributes. Addition to this, the engineers also need to have security knowledge and understand the background and standard related to security for the business requirements. These constraints resulted to eliciting poor quality security requirements that leads to insecure application development. In this paper, we propose our elicitation approach for security requirements by using Essential Use Cases (EUCs) and describe an example usage of eliciting security requirements for Internet of Thing (IoT) applications by using an Essential Use Case (EUC). It is found that EUCs to enhance the process of eliciting security requirements to produce accurate and complete security requirements for IoT applications.

Key words : Eliciting, Internet of Things (IoT), IoT Applications, Security Requirements, Essential Use Case (EUC), Essential User Interface (EUI).

1. INTRODUCTION

Requirement engineering (RE) in general can be used to solve the requirements issues and play a prime help for succession of the software engineering projects. RE is concerned with the process of defining, documenting and maintaining of the requirement. Moreover requirement engineering contains a set of activities such as: requirements inception or requirements elicitation, requirements identification, requirements analysis, requirements specification, system modeling, requirements validation and requirements management [1]. The most significant part in developing an application or software is requirement elicitation in the early phase of the development. Requirement elicitation is extract and determined the requirement from stakeholders, through the elicitation techniques. In addition, the requirements include both functional and non-functional requirements. Thus, the requirement elicitation is the basic part of requirement engineering to assure the success of the software development with high quality. Problems in understanding result from the necessary involvement of requirements analysts, sponsors, developers, and end users in requirements elicitation. The requirements are produced and interpreted by people with different experience levels and backgrounds. The form in which the requirements are expressed and the size of the system described by the requirements also affect understanding. If the participants in elicitation do not adequately understand the output of the process, then the resulting requirements may be ambiguous, inconsistent, and incomplete. The requirements may likewise be mistaken, not tending the true needs of the elicitation networks.

The number of IoT devices are expected to increase to 50 billion in the year 2020 [2]. If the number of connected devices increases and the management table size and other management information exceed the system capacity, the performance of the service may deteriorate or the service may become uncontrollable. This could become a serious problem, in particular, if the IoT devices and applications are related to the life of a person, property, or social infrastructure. Therefore, when designing the size of the management table and other system capacity-related settings, it is necessary to consider the expected increase in the number of connected devices. In the development of IoT application such as logistic and lifetime management, agriculture and breeding, smart mobility and smart tourism, smart grid, smart building and many others [3][4], security requirements are very important. It is because in today's world of daily virus alerts,

malicious crackers and various other threats of cyber terrorism it is very difficult to make the application development successful. Eliciting security requirements is very difficult because there is no well-defined process for eliciting security requirements and because requirement engineer is trained to elicit functional requirements and some non-functional requirements but not security requirements. Most requirements engineers are poorly trained to elicit, analyze, and specify security requirements, often confusing them with the architectural security mechanisms that are traditionally used to fulfill them [5]. Thus, the requirement engineer ends up specifying architectural and behavioral constraints rather than actual security requirements. To elicit these security requirements the engineer must have a clear understanding of various types of security requirements such as, authentication, confidentiality, integrity, authorization, access control, and availability [3] etc. Likewise, we feel that there must be a well-defined process for eliciting security requirements so that there is no other constraint for the design and implementation team of the application.

In this paper, we present an approach to elicit security requirements using EUC and EUI. This paper organized as follows: Section 2 present the background and motivation. Section 3 presents the example of the usage for IoT application using EUC and EUI. Section 4 concludes the paper with some discussion about security requirements and future works.

2. BACKGROUND AND MOTIVATION

2.1 Security Requirements Elicitation

Security requirements are reflected in various nontechnical security controls that address such matters as policy and procedures at the management and operational elements within organizations, again at differing levels of detail. It is important to define the context for each use of the term security requirement so the respective communities (including individuals responsible for policy, architecture, acquisition, engineering, and mission/business protection) can clearly communicate their intent. Meanwhile, requirements elicitation is a process of deriving the system requirements through observation of existing systems, discussions with potential users and procurers, task analysis, and so on. This may involve the development of one or more system models and prototypes. These help users understand the system to be specified [1]. Numerous requirements engineering research and practice have tended the capabilities that the system will provide. So a great deal of consideration is given to the functionality of the system, from the user's perspective, but little consideration is given to what the on system should not do [6]. In one discussion requirements prioritization for a specific large system, ease of use was assigned a higher priority than security requirements. Security requirements were in the lower half of the prioritized requirements. This happened in part because the only security requirements that were considered had to do with access control. However, the basic purpose of eliciting security requirements is to protect the software or applications. A software system means along with the software it includes an operating environment, in which the software runs, the physical environment, in which the system exists, the people interacting with the system directly or indirectly, and other systems. For ensuring such system security requirements need to be defined.

The requirements elicitation and analysis that is needed to get a better set of security requirements seldom takes place. Even when it does, the security requirements are often developed independently of the rest of the requirements engineering activity and hence are not integrated into the mainstream of the requirements activities. As a result, security requirements that are specific to the system and that provide for protection of essential services and assets are often neglected. In the real world of the current practice of security guidance and solution, most developers or engineers refer to the Common Criteria (CC), although the CC is complex and difficult to understand by novice [7] [8]. They found that most of the developers tend to make mistakes when determining the right security requirements. They proposed a method that allows users to specify security requirements, which subsequently allows developers to specify the security attributes of their products; hence, allowing evaluators to determine if the products actually meet their claims. However, the developer could not specify the security attributes at the early requirements phase of product development. Further, eliciting security requirements at the early phase of development is needed and it is crucial to present the secure software or applications.

S. Yahya et. al [9] had explore the usage of Essential Use Case (EUC) to capture the security requirements from the business requirements. The study evaluates various security requirement-engineering tools and analyses the existing gaps in security requirement engineering tools. They have conducted a review of seven common security requirement-engineering tools to identify the gaps and problems that are still outstanding. They found that there are several works done using semi-formalized model, but almost none of the work captures the security requirements from the textual representations especially by using EUC.

2.2 Essential Use Case (EUC) and Essential User Interface (EUI)

The EUC is defined by Constantine and Lockwood [10] as a "structured narrative, expressed in

a language of the application domain and of users, comprising a simplified, generalized, abstract, technology free and independent description of one task or interaction that is complete, meaningful, and well-defined from the point of view of users in a role or some roles in relation to a system and that embodies the purpose or intentions underlying the interaction". The main objectives are to support better communication between the developers and stakeholders via a technology-free model and to assist better requirements elicitation. These objectives can be achieved by allowing only specific details relevant to the intended design to be elicited [11].

An EUC is shorter and simpler compared to a conventional use case as it comprises an abstraction of only essential steps and the user's intrinsic interest. It comprises just user intentions and system responsibilities permitting users to capture the core part of the requirement without the need to describe the user interface in detail. An EUC aims to identify "what the system must do" without being concerned on "how it should be done". These questions often lead to critical realizations that allow users to rethink, or reengineer the aspects of the overall business process.

Figure 2.1 demonstrates an example of text requirements (left hand side) and an example of an EUC (right hand side) while capturing the requirements (adapted from [12]). The text requirements from which the important phrases are extracted (highlighted) are shown on the left hand side. From the text requirements, a specific key phrase (essential requirement) is abstracted and is shown in the EUC on the right-hand side of Figure 2.1. The EUC depicts two interrelated sets of information: the user intentions and system responsibility as shown in Figure 2.1. An EUI prototype is a type of abstract prototype or paper prototype that is a low-fidelity model. Also known as a "UI prototype" for a software system, it represents the general ideas rather than the exact details of the UI [12][13]. An EUI prototype represents the user interface requirements in a technology independent manner; just as the EUC models do for the behavioral requirements. An EUI prototype is particularly effective during the initial stages of user interface prototyping for a system. It models user interface requirements that are evolved through analysis and design to the final user interface of a system [13]. It also allows some investigation of the ease of use of a system.

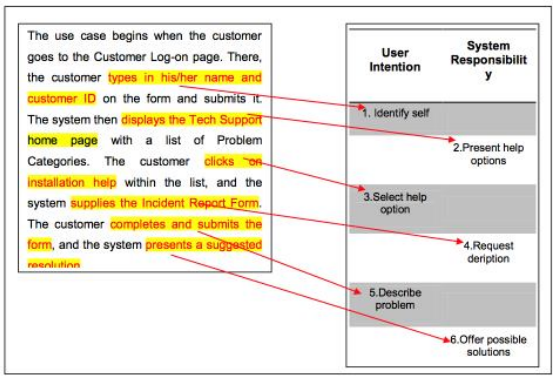

Figure 0.1: Example of Textual Requirement (left) and Example of Essential Use Case (EUC) (right)

When capturing requirements from textual requirement, the EUC model is found to be more reasonable than the conventional Unified Modelling Language (UML) use case. An equivalent EUC description is generally shorter and easier than a conventional UML use case as it just includes the basic steps (core requirements) of user's intrinsic interest. It contains the user's intentions and the system responsibilities to document the specific interaction without the need to describe the user's interface in detail. It is reported in [14][15] that EUC are beneficial for capturing security requirements.

EUI prototyping is a low fidelity prototyping approach [16]. It provides the general idea behind the UI instead of its exact details. Focusing on the requirements rather than the design, it represents UI requirements without the need for prototyping tools or widgets to draw the UI [17]. EUI prototyping extends from and works in tandem with the semi-formal representation of EUC that also focuses on the users and their usage of the system, rather than the system features [18]. It thus helps to avoid clients and REs from being misled or confused by chaotic, evolving and distracting details. EUI also allows some explorations of the usability aspects of a system. Figure 2.2 shows examples of EUI prototype developed from EUC models.

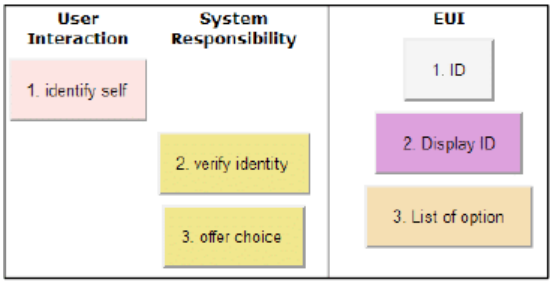

Figure 0.2: Examples of EUI Prototype from EUC Models

3. EXAMPLE USAGE

IoT has the capabilities to make homes and our life smarter. It consists of different supporting

technologies. Technologies help people in many ways, for example, a mobile application assist travelers to travel via LRT using PDA and GPS [19]. Therefore, people percentage having access to technologies is rapidly increasing and need to have access to information anytime, anywhere. These innovative technologies provide convenience in everyday activities, energy efficiency, security, and comfort [20]. Adding intelligence capabilities to various IoT industry could provide increased life quality for the sick and elderly, for example. Much of the attention in research has revolved around wireless technologies that are supportive of remote data control, sensing, and transfer, such as mobile networks, RFID, Wi-Fi, and Bluetooth, [3] which have been used to embed intelligence into the environment. Our previous study [21] has demonstrate the use of IoT security library to elicit security requirements based on Smart Parking System scenario. Table 1 shows the EUC and EUI for Smart Parking.

Textual Requirement:

User need to book the parking area by using mobile applications. The user is required to enter ID of the required parking area. User finds a parking area from the list of area, registered by parking details. The system will find the shortest path and sent the information to the user. The system gives the details of the selected parking areas such as the name, price per minute, number of total available slots. User finds the location selected by using GPS. Then, the GPS location updated to a cloud server. After that, user enter the parking. The system read the RFID tag and all the information updated to cloud and to neighbor car park. After the user exit the parking, again RFID tag will read and updated to cloud. Lastly, billing information will send to the user after checkout.

Table 1: EUC and EUI for Smart Parking

Essential Use Case (EUC)		**Essential User Interface (EUI)**	**Attributes**	**IoT Security Requirements**	**IoT Technologies**
User Interaction(UI)	**System Responsibility (SR)**				
1. Enter ID		1. Input ID	Username Password PIN ID card	Authentication	
	2. Verify user ID	2. Display ID	Permission Verify Gain access	Authorization	
3. Update location and type of vehicle		3. List location and type of vehicle	Accessible Obtainable Software patching	Availability	
	4. Verify location	4. Display location	Permission Verify Gain access	Authorization	
	5. Verify type of vehicle	5. Display type of vehicle	Permission Verify Gain access	Authorization	
6. Enter parking detail		6. Input detail	Username Password PIN ID card	Authentication	
	7. Finds the shortest path and sent the info to user	7. Display shortest path	Accessible Obtainable Software patching	Availability	

8. Find the location		8. Submit location	Limits access Unreadable data Restricted access	Confidentiality	
	9. **Mobility Networks (GPS)** location updated to cloud server		North coordinate East coordinate Altitude Signals		GPS
	SubEUC (1)				
	UI	**SR**			
	Check GPS coordinate		9. Notify GPS location	Permission Verify Gain access	Authorization	
		Show coordinate		Accessible Obtainable Software patching	Availability	
	Update location			Limited access Control the access	Access Control	
10. Receive GPS location		10. Display GPS location	Limited access Control the access	Access control	
11. Enter the car park		12. Input detail	Username Password PIN ID card	authentication	
	13. **RFID** tag is read and authenticated by RFID reader		RFID tags/ transponder RFID readers		RFID
	SubEUC (2)				
	UI	**SR**			
	Tag read the code and ID		13. Verify ID using RFID	ID tag	Authentication	
		Verify code and ID		Permission Verify Gain access	Authorization	
		Request information		Limits access Unreadable data Restricted access	Confidentiality	
	Receive information			Limited access Control the access	Access Control	
	14. Info updated to cloud and to neighbour car park		14. Display status	Accessible Obtainable Software patching	Availability	

<table>
<tr><td>15. Exit the car park</td><td colspan="2"></td><td>15. Notify system</td><td>Protect data
Unmodified data
Unaltered data</td><td>Integrity</td><td></td></tr>
<tr><td rowspan="7"></td><td colspan="2">16. RFID tag is read by RFID reader and updated to cloud</td><td rowspan="7">16. Verify ID using RFID</td><td>RFID tags/ transponder
RFID readers</td><td></td><td>RFID</td></tr>
<tr><td colspan="2">SubEUC (3)</td><td></td><td></td><td></td></tr>
<tr><td>UI</td><td>SR</td><td></td><td></td><td></td></tr>
<tr><td>Tag read the code and ID</td><td></td><td>Username
Password
PIN
ID card</td><td>Authentication</td><td></td></tr>
<tr><td></td><td>Verify code and ID</td><td>Permission
Verify
Gain access</td><td>Authorization</td><td></td></tr>
<tr><td></td><td>Request information</td><td>Limits access
Unreadable data
Restricted access</td><td>Confidentiality</td><td></td></tr>
<tr><td>Receive information</td><td></td><td>Limited access
Control the access</td><td>Access Control</td><td></td></tr>
<tr><td></td><td colspan="2">17. Billing info is sent to the user.</td><td>17. Submit billing info</td><td>Permission
Verify
Gain access</td><td>Authorization</td><td></td></tr>
</table>

4. CONCLUSION AND FUTURE WORK

Requirements Engineering research has been established for many years and requirements are the core part of any application or system development. It is found that requirements written in natural language are error-prone, incorrect and incomplete. Complexity in security requirements requires requirements engineers to have a security experience and knowledge in the process of eliciting and analyzing requirements. Due to these deficiencies, semi-formal models such as EUC have been developed to improve the quality of the requirements as well as to minimize the time taken and to ease the requirements capturing process. However, almost none of the work explores the usage of EUC for capturing security requirements. This study attempts to investigate the usage of EUC to improve the process of capturing the security requirements for IoT applications. Thus, we believe this study will positive impacts to the IoT industry as it contributes to the improvement of developing secure IoT applications, hence leading to the better quality security requirements. For future work, we will work on the possibility of automating the complicated usage of the standard to the easier and simpler practice by using our tool support. In addition, this approach creates the possibility to allow a consistent and complete eliciting proses of the security requirements from IoT applications. Thus, the focus is to provide the end-to-end support for both the requirement engineers and the client in confirming the consistency of requirements.

ACKNOWLEDGEMENT

The authors would like to acknowledge Universiti Teknikal Malaysia Melaka (UTeM) and Ministry of Education (MoE) for its support and the funding of this FRGS research grant: FRGS/1/2016/ICT01/FTMK-CACT/F00325.

REFERENCES

1. I. Sommerville, ***Software Engineering Ninth Edition,*** Ninth. Boston, Massachusetts: Person Education, Inc., Addison-Wesley, 2011.
2. S. Morisaki, **Guidance for Practice Regarding 'IoT Safety / Security Development Guidelines**, 2017.
3. A. A. Ibrahim and M. Kamalrudin, **Security Requirements and Technologies for The Internet of Things (IoT) Applications: A Systematic Literature Review,** *J. Theor. Appl. Inf. Technol.*, vol. 96, no. 17, pp. 5694–5716, 2018.
4. E. Borgia, T**he Internet of Things Vision: Key Features, Applications and Open Issues**, *Comput. Commun.*, vol. 54, pp. 1–31, 2014.
5. D. G. Firesmith, **Engineering Security Requirements**, *J. Object Technol.*, vol. 2, no. 1, pp. 53–68, 2003. https://doi.org/10.5381/jot.2003.2.1.c6
6. P. Salini and S. Kanmani, **A Survey on Security Requirements Engineering**, *Int. J. Rev. Comput.*, vol. 8, no. December, pp. 1–10, 2011.
7. E. Paja, F. Dalpiaz, M. Poggianella, P. Roberti, and P. Giorgini, **STS-Tool : Socio-technical Security Requirements through Social Commitments**, pp. 331–332, 2012.
8. M. S. Ware, J. B. Bowles, and C. M. Eastman, **Using the Common Criteria to Elicit Security Requirements with Use Cases**, in *Proceedings of the IEEE, Memphis, TN, USA*, 2006, pp. 273–278.
9. S. Yahya, M. Kamalrudin, and S. Sidek, **A Review on Tool Supports for Security Requirements Engineering**, in *2013 IEEE Conference on Open Systems, ICOS 2013*, 2013, pp. 190–194.
10. L. L. Constantine and L. A. . Lockwood, ***Software for Use : A Practical Guide to the Models and Methods of Usage-Centered Design*.** New York: ACM Press, Inc/Pearson Education, Inc., 1999.
11. R. Biddle, J. Noble, and E. Tempero, **Essential Use Cases and Responsibility in Object-Oriented Development**, *J. Aust. Comput. Sci. Commun.*, vol. 24, no. 1, pp. 7–16, 2002.
12. L. L. Constantine and L. A. D. Lockwood, **Structure and Style in Use Cases for User Interface Design**, vol. 1, no. 978. Addison-Wesley Longman Publishing Co., Boston, MA, 2001.
13. S. W. Ambler, **Essential (Low Fidelity) User Interface Prototypes**, 2003. .
14. S. Yahya, M. Kamalrudin, S. Sidek, and J. Grundy, "Cases (EUCs)," in *Asia Pacific Requirements Engineering Symposium (APRES) 2014*, 2014, pp. 16–30.
15. M. Kamalrudin, J. Hosking, and J. Grundy, **MaramaAIC: Tool Support for Consistency Management and Validation of Requirements**, *Automated Software Engineering*, pp. 1–45, 2016.
16. S. W. Ambler, **Essential (Low Fidelity) User Interface Prototypes**, 2003.
17. L. L. Constantine and L. A. D. Lockwood, **Usage-Centered Software Engineering : An Agile Approach to Integrating Users , User Interfaces , and Usability into Software Engineering Practice 2. Agile Methods and Usability**, in *25th International Conference on Software Engineering, Portland, Oregon, 2003*, 2003, pp. 746–747.
18. S. W. Ambler, **The Object Primer: Agile Model-Driven Development with UML 2.0** (3rd ed.), 2004.
19. A. M. H. Lim, A. A. Azlianor, S. M. Suhaizan, and K. Massila, **Architecture of Mobile Web Application for Generating Dynamic Route Map**, *Int. J. Comput.*, vol. 2, no. 2, 2008.
20. E. Alsaadi and A. Tubaishat, **Internet of Things : Features, Challenges, and Vulnerabilities,** *Int. J.*

Adv. Comput. Sci. Inf. Technol., vol. 4, no. 1, pp. 1–13, 2015.

21. M. Kamalrudin, A. A. Ibrahim, and S. Sidek, **A Security Requirements Library for the Development of Internet of Things (IoT) Applications**, in *Requirements Engineering for Internet of Things*, vol. 809, 2018, pp. 87–96. https://doi.org/10.1007/978-981-10-7796-8_7