[1]Asma Asdayana Ibrahim,
[2]Massila Kamalrudin

SecIoT_MEReq : Automated Tool To Elicit Security Requirements For IoT Application Software Development

Journal of Electrical Systems

Abstract:
The use of Internet of Things (IoT) applications has grown in popularity and significance because it allows people and services to interact at any time and from any location. IoT will necessitate the development of new software tools as well as interoperability: Until now, the Internet of Things opportunity has consisted of "simple" monitoring applications and related tracking or location services. Security has long been regarded as a major concern in IoT. It is critical for IoT application developers to elicit security requirements of IoT applications at an early stage to avoid potential security issues. In this paper, we describe our automated approach and tool, called SecIoT_MEReq that helps to elicit security requirements of IoT applications. SecIoT_MEReq provides a model-based approach together with patterns library that helps to capture requirements that have been expressed in textual natural language requirements then extracted to Essential Use Cases (EUCs) and Essential User Interface (EUI) models. We describe its design and implementation together with the results of evaluating our tool's usability. The results of the study showed that our tool can help requirements engineers to easily elicit security-related requirements of IoT applications development.

Keywords: security requirement, Internet of Things, IoT technologies, elicitation, IoT applications, EUC, EUI

1 INTRODUCTION

Internet of Thing (IoT) applications have been used widely as they allow interactions between people and things anywhere and anytime. The use of IoT application is rapidly growing, especially in performing industrial domain, smart city domain and health well-being domain [1]. There is also a plethora of applications being developed to fulfil the needs of IoT application users. Regulation surrounding IoT is in the early stages or non-existent, and is inconsistent across geographies. Security must be the factored in from the beginning of development of any IoT product or application. Security worries among businesses and consumers are driving an increased interest in and need for government involvement. Many user consider the IoT opportunity exciting, but many still feel they are lacking the necessary technology, skills or tools [2]. Some of developers do not have or are unsure if they have the necessary technology today to deliver on IoT expectations. They are unsure or definitely do not have the necessary skills and resources today to deliver on IoT expectations. Therefore, there is a need for automation support to capture and elicit security related requirements at the early stage of IoT application requirements engineering. In our previous work [3], we conducted a study of common practices of involvement in security requirements elicitation among practitioners in the area of IoT. The results of the study shows that the professionals have knowledge and security training, but they did not know how to use and handle it in the earlier phase of application developments. The survey also indicated there is a lack of a complete set of standard or solutions for eliciting security requirements that can be applied during the process of applications development in order to achieve quality and secure applications. This study also found that the respondents used multiple solutions in handling the security issues rather that considering one solution only. Therefore, these challenges have motivated us to:

i) Develop an automated tool support for eliciting security requirements
ii) Evaluate the tool to demonstrate its ability to enhance the correctness and usability for eliciting security requirements of IoT applications.

This paper describes the approach and an automated tool that captures and elicits security requirements for IoT applications using Model-Driven Development (MDD) with semi-formalised model, Essential Use Cases (EUCs) and Essential User Interface (EUI). We present background for this study, our prototype tool, and then we

[1]Politeknik Sultan Azlan Shah, 35950, Behrang, Malaysia
[2]Universiti Teknikal Malaysia Melaka, 76100, Malaysia
Email: asdayana@psas.edu.my,
massila@utem.edu.my (corresponding author)

conducted evaluations to test and validate the efficacy of the developed approach and tool. Finally, we discuss related works and future work.

2 RESEARCH BACKGROUND AND RELATED WORKS

IoT applications become an increasingly attractive target for cybercriminals. This will require security engineers to work closely with the developers of the IoT capability to introduce security requirements early in the design process. More connected devices mean more attack vectors and more possibilities for hackers to target us. Some of the more frightening vulnerabilities found on IoT devices have brought IoT security further up the stack of issues that need to be addressed quickly. The researcher found critical vulnerabilities in a wide range of IoT devices and applications, which could be leveraged by hackers to carry out several malicious activities, including monitoring live feeds, changing camera settings and authorizing other users to remotely view and control the monitor. As defined by [1] vulnerabilities including poor encryption and backdoors that could allow unauthorized access have been found. With any physical device, there is a chance that a hacker could manipulate it and get into exposed USB ports or a debugger interface, if someone is able to unsuccessfully hack at the embedded level into an IoT device's memory and can be read the encryption key, every device that is or has been shipped become vulnerable. The network is only as strong as its weakest link.

In another development, it was proven that Internet-connected cars can be compromised, as well, and hackers can carry out any number of nefarious activities, including taking control of the entertainment system, unlocking the doors or even shutting down the car in motion [2]. So, security is a very serious issue in IoT applications development. As applications devices with low quality or improper security are released into the market, people may soon lose trust in the application and its devices that enter the market. Before being able to secure a system, it is important to first understand the functional and technological details of the system to be secured. This will require security engineers to work closely with the developers of the IoT capability to introduce security requirements early in the design process. Using the methodical systems security engineering approach for each IoT implementation within an enterprise is recommended. Information security, privacy, and data protection systematically be addressed at the design stage. Unfortunately, in many cases, they are added on later once the intended functionality is in place. This is not only limits the effectiveness of the added on information security and privacy measures but also is less efficient in terms of the cost to implement them. However, the IoT objects do not always have enough computing power to implement all relevant security layer functionalities, the heterogeneity of objects become very challenging in this context. Similarity, the heterogeneity of privacy policies needs to be taken into account.

Meanwhile, Alsaadi & Tubaisat [1] mentions that there is lack of standards for authentication and authorization of IoT edge devices. Some of IoT devices have no authentication capabilities while others have limited support. Very few have capabilities that support multi-factor authentication. Although some standards or commercial options are available, for example, certificate authentication, commercial or semi-commercial identity providers such as Google, there is lack of ability to create device-specific profiles and authorization options and the privacy implications of using these services providers has not been fully explored. Every single device and sensor in the IoT represent a potential for risk.

In addition, with more connected devices mean more attack vectors and more possibilities for hackers to target us. Wearable also can become a source of threat to our privacy, as hackers can use the motion sensors embedded in smart watches to steal information that we are been typing, or they can gather data from smart watches application or health tracker devices you might be using. Some of the most worries cases of IoT hacks involves medical devices and can have detrimental and perhaps fatal consequences on patient health. Only by ensuring the security, IoT can be more universal, so there is need to strengthen the security of the IoT. Based on recent reports, the number of connected 'things' is set to explode and expected to reach 100 billion by 2025 [3] [7]. With a rapid increase in the demand for IoT applications, securing the information content delivered among various entities involved in IoT applications development has become an important issue. Therefore, there is a need to develop an approach or tool to secure the development of IoT applications in order to achieve quality and secure applications.

Abraham [15] creates the GARMDROID tool. The tool was designed to help IoT software developers and integrators assess IoT security issues using the visualisation of hardware requests from Android applications.

The Android Asset Packaging Tool (AAPT), which is a component of the platform tool set, is the foundation upon which GARMDROID is built. Clients could send malware samples and ask for analysis over a Web interface in this execution. This process is dependent on a static review of the permissions that Android applications request. During the analysis, GARMDROID uses a series of bash and python scripts to instruct AAPT to extract the contents of the app's AndroidManifest.xml file and to filter the crucial strings. This happens once an android application file (.apk) is uploaded by a user. Every time, in addition to the collection of permissions requests, GARMDROID shows an inference of the hardware characteristics sought by the programme being examined. These examples serve two purposes: to demonstrate how GARMDROID works and to focus the conversation on observations that may help discover security risks in Android applications geared toward the Internet of Things. The security requirements for IoT applications are not covered by this tool, which only assesses IoT security concerns upon viewing of an Android application hardware request.

On other work, Dhillon & Kalra [5] create an AVISPA tool as part of their other work. It was created as a push-button tool for examining complex Internet security protocols and applications. The protocols are written in the HLPSL coding language (High Level Protocol Specification Language). HLPSL consists of basic roles that represent various participants as well as roles that are composed to reflect basic role scenarios. Each role operates independently of the others, gathering some initial data via parameters and corresponding with other roles via channels. This paper suggests a simple key agreement and remote user authentication system based on biometrics for secure access to IoT services. The protocol employs XOR and lightweight hashing techniques. It is resilient against a variety of security assaults, according to the security analysis. The AVISPA tool is used for the formal verification, which validates its security in the presence of a potential attacker. The four phases of the proposed multifactor biometric user authentication are the phases of user registration, login, authentication, and password change. The suggested protocol is extremely ideal for the resource-constrained IoT devices because it only makes use of one-way hash, perceptual hash functions, and XOR operations, which are computationally less expensive. It is resilient against a variety of security assaults, according to the security analysis. The AVISPA tool is used for the formal verification, which validates its security in the presence of a potential intruder. The elicitation and analysis of IoT applications in the early stages of development are not, however, covered by this work.

Meanwhile, ElicitO is a tool that supports a standardized quality terminology and ontological constructs to capture NFRs throughout the RE activities [6]. ElicitO also supports knowledge-based reasoning methods, enabling the semi-automation of RE tasks like conflict detection during priority setting and authenticity verification during requirements validation. The tool contains two levels in total: The ontology layer is the top layer, and in the Protege database, the quality and domain ontologies are encoded as OWL components. Under application and GUI components, the user communication layer, which is the second layer, degrades. When the Protege API makes a request for domain knowledge and the associated quality attributes, the application component interacts with the ontology layer. The graphical user interface is used to present the user with all query results and information. MySQL serves as the underlying database for storing a requirements session. This application supports RE efforts by utilising knowledge management strategies and high-quality ontologies. A consistent vocabulary to handle quality concerns/aspects across RE activities is provided by the ontology, which implements the quality measurements and attributes outlined by the ISO/9126 quality model. Through a case study involving the creation of an intranet portal project at the University of Manchester, they accept how the framework and tool can be utilised to help the requirements elicitation and prioritisation activities in an effective manner based on their observations. As part of its knowledge-based system capabilities, ElicitO also codifies the quality model standard into automated ontologies, assisting in the semi-automation of RE tasks. Additionally, ElicitO provides constructs that allow for the incorporation of quality concerns throughout the early phases of software engineering as well as the explicit modelling of links between functional and NFRs. However, this tool is employed to record NFRs during RE efforts. It cannot be used as a method to elicit security requirements. Additionally, ElicitO does not offer security mechanisms for IoT applications and cannot be used across many projects or apps by a range of stakeholders.

The Heuristic Requirements Assistant (HeRA) [7] is a tool that applies security-relevant heuristics to requirements and service descriptions in order to identify possible security issues. HeRA raises awareness and provides feedback while people write requirements. The HeRA tool supports technical experts, as well as security experts, in identifying potential security issues. HeRA is integrated with the CC-based requirements method and

together these two techniques make up the elicitation phase of SecReq. SecReq is a security requirements elicitation and tracing methodology based upon the methodology connected toward ETSI. SecReq enhances the ETSI methodology with security requirements elicitation and writing support, as well as requirements analysis and tracing capabilities. The added elements are supported by a systematic use of different sources of security expertise and experience and by integrating three existing techniques, namely the CC (from the ETSI methodology), the heuristic requirements editor HeRA, and the model-based security engineering approach UMLsec. Furthermore, HeRA provides a requirements editor who allows technicians to enter the system functional information, for example the service requirements. The input to this editor is checked against security-related heuristics. In particular, these heuristics search for keywords and patterns that may indicate security-relatedness. This search for security keywords is in SecReq used, among other things to help a developer in selecting appropriate parts of the CC security requirements knowledge: Thus, HeRA works closely together with the CC-based method. However, this tool does not offer elicitation of security requirement of IoT application.

Gope & Hwang [8] created a BSN-Care to secure IoT-based healthcare system, using BSN (Body Sensor Network). One of the key IoT breakthroughs in healthcare is the use of BSN technology, which enables a patient to be monitored via a network of wireless sensor nodes that are light and small-powered. Patients' privacy is put at risk when new technology is developed for healthcare applications without taking security into account. LPU is crucial in the system that is being suggested. The sensor data is gathered and securely sent to the BSN-Care server. All of the fundamental security needs of an IoT-based healthcare system can be satisfied by the BSN-Care system. However, this tool does not suggest how to elicit and analyse security requirements for other IoT-based applications; it solely monitors security in the healthcare system.

The Haier SmartCare is a smart device made to manage and read data from several sensors positioned all over a user's home, including a smoke detector, water leakage sensor, sensor to determine whether the doors are open or closed, and sensor to determine whether the remote power is switched [9]. Through the ZigBee protocol, these sensors are linked. The major purpose of this gadget is to provide customers with the ability to more effectively monitor their houses while they are away and to receive alerts based on sensor data. They employ both commercial and industrial IoT devices, from which the security of hardware, software, and networks is studied and backdoors are identified, to better understand the security flaws of current IoT devices and to encourage the creation of low-cost IoT security approaches. A thorough security analysis technique performed on a smart metre and a home automation system demonstrates that most devices frequently have security flaws. To assist IoT product manufacturers in securing their offerings, security solutions and mitigation techniques are explored. The chosen sample IoT devices comprise a smart controller for a home automation system and a smart metre for contemporary power networks, assuming that IoT devices are widely employed in both business and industrial applications. They validate and illustrate the shortcomings of current IoT device design techniques while defending against various cyber-attacks from the hardware, software, and network layers through these assessments. In order to deploy more secure devices in the upcoming IoT age, they continue to create remedies to reduce security concerns for existing IoT devices. This technology does not cover eliciting security criteria; rather, it is used to monitor a house and receive alarms using sensor data.

3 MODEL-BASED APPROACH

3.1 Model-Driven Development (MDD)

Model-Driven Development (MDD) is a software engineering approach that uses model to create a product. MDD is sometimes used interchangeably with model-driven engineering, and may refer to specific tools and resources, or a model-driven approach [4] [5]. MDD is part of a trend toward more diverse approaches to the development of IT products. Another aspect of this innovation is agile practices, which is in some cases are associated with model-driven development. Ideas about the development and engineering process now play a major role in IT processes, especially in larger companies where a more detailed staff hierarchy adds more layers to a process.

MDD has emerged as one of the leading approaches for enabling rapid, collaborative application development [6]. Because model-driven development uses visual modelling techniques to define data

relationships, process logic, and build user interfaces, model-driven software development empowers both developers and business users to rapidly deliver applications without the need for code [7] [5].

Atkinson & Kühne [8] refine the simple one-size-fits-all view of instantiation and adopt a more sophisticated view of metamodeling's role in MDD. This method places ontological instances of relationships, which connect user concepts to their domain types, in a secondary role. In other words, linguistic instance-of relationships cross (and form the basis for) linguistic metalevels, whereas ontological instance-of relationships do not; they relate entities within a given level. This is the new UML 2.0 and MOF 2.0 standards' interpretation of the four layer architecture. Although the latter take a primarily linguistic perspective, it is useful to allow ontological (intralevel) instantiation to establish its own type of ontological (vertical) metalevel boundaries.

An organization's desire to increase the return on its software development investment is the primary driver behind MDD. It accomplishes this in two main ways: By expanding the functionality that a primary software artefact offers, it raises the short-term productivity of developers. It increases long-term productivity for developers by slowing the rate at which a key piece of software ages. These requirements make it obvious that visual modelling is one of the technological pillars of MDD assistance. Software can be written and implemented using the MDD format, which is quick, efficient, and inexpensive.

For our study, we develop SecIoTA Model that comprises the need of 1) security requirements and 2) IoT technologies to develop a secure IoT application. To develop a secure IoT application, requirements that almost needed are security requirements and its technologies. The security requirements that most needed are authentication, authorization, availability, confidentiality, access control, and integrity while IoT technologies that most used are sensor, mobility network, RFID system, Bluetooth and Wi-Fi. Combination of this two requirements may help the developer to develop and design a secure IoT application in the future. We use this model to our approach to create our product and realise our approach using an automated approach. We also use semi-formalised model using EUC and EUI.

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

An EUC is a structured narrative that is expressed in the users' and the application domain's language. It entails the description of a single action or interaction in a compressed, abstract, technology-free, and independent of implementation form. [8][9]. From the perspective of the users, an EUC is a comprehensive, meaningful, and well-designed encounter. It embodies the goal or goals underlying the interaction and represents a specific role in regard to a system. By removing the influence of implementation choices, EUCs allow users to pose fundamental questions like "what's really happening" and "what do we really need to do." These inquiries frequently result in crucial insights that empower users to reevaluate or reengineer various parts of the overall business process. While capturing the needs, Figure 1 displays an example of natural language requirements on the left and an example of an EUC on the right (adapted from [13]). On the left side of Figure 2, you can see the natural language requirements from which the key terms are extracted (highlighted). A unique key phrase (important requirement) is abstracted from the natural language requirements and is displayed in the EUC on the right-hand side of Figure 1. The user intents and system responsibility are two interrelated types of information that are represented by the EUC, as shown in Figure 2.

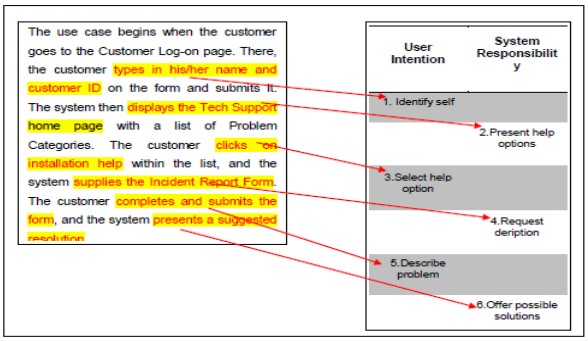

Figure 1: Example Natural Language Requirement and Example of EUC

A paper prototype, abstract prototype, or low-fidelity model is what is known as an EUI prototype. For a software system, it is also referred to as a "UI prototype", and it displays the broad concepts rather than the precise elements of the UI [9][10]. Similar to how EUC models describe behavioural requirements, an EUI prototype represents user interface requirements in a way that is independent of technology. When designing a system's user interface in its early stages, an EUI prototype is especially useful. It simulates the user interface requirements that develop into the system's ultimate user interface through analysis and design [10]. It also enables some investigation into a system's usability features. An example of an EUI prototype created from an EUC model is shown in Figure 2. The user intention/system responsibility dialogues are used to capture the potential high-level UI functionality.

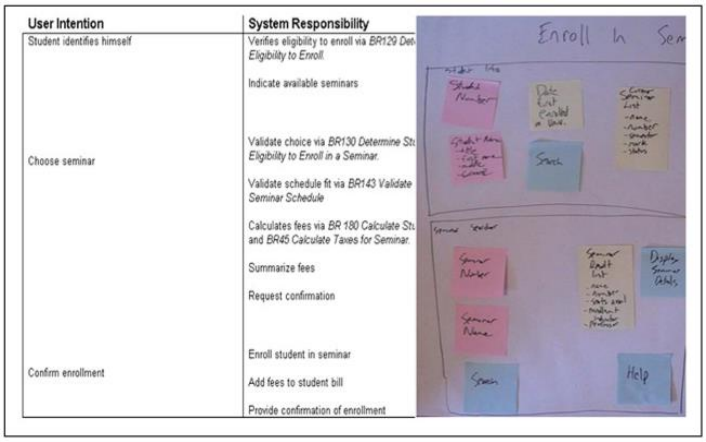

Figure 2: Example of EUI Prototype Iterates from EUC Model

4 AUTOMATED TOOL: SECIOT_MEREQ

The purpose of this study was to develop an approach and an automated tool to assist requirements engineers to automatically capture and elicit the security-related requirements of IoT application. We developed an automated tool, SecIoT_MEReq, that automatically elicit the security requirements of IoT application. Our approach for requirements elicitation employs the Model-Driven Development (MDD) using semi-formalised models, Essential Use Cases (EUCs) and Essential User Interface (EUI). MDD emphasizes the use of models at a higher abstraction level in the software development process and argues in favour of automation via model execution, transformation, and code generation [17]MDD promotes reuse at the domain level, improves quality through incremental model enhancements, lowers costs through the use of an automated process, and lengthens the useful life of software solutions.

Therefore, this research proposes a new model-based approach to support requirements engineer in eliciting security requirements for secure IoT applications development. Figure 3 below shows the new approach of this research.

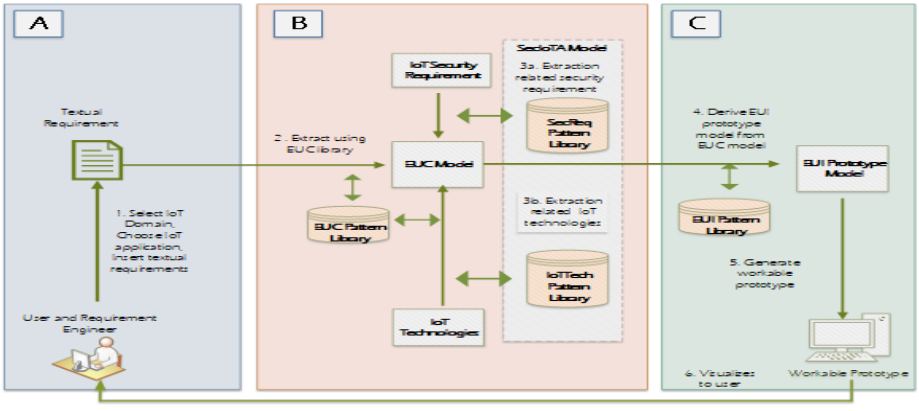

Figure 3: An overview of Model-based Approach for Eliciting Security Requirements

In Step 1, textual requirements will be collected from user and requirement engineers. To begin, they choose an IoT domain: (1) industrial domain, (2) smart city domain, and (3) health well-being domain. Second, they select an IoT application. The industrial domain, for example, is divided into three domains: I logistic and product lifetime management, ii) agriculture and breeding, and iii) industrial processing. Smart cities are divided into four domains: I smart mobility and smart tourism, ii) smart grid, iii) smart homes/buildings, and iv) public safety and environment monitoring. Finally, the domain of health and well-being is subdivided into two domains: I medical and healthcare, and ii) independent living and inserted textual requirements. Lastly, they included a textual requirement as a scenario from the IoT application. In Step 2, to generate an EUC model, the requirements are analysed and extracted using the Essential Use Case (EUC) pattern library. Using keyword matching, the searching process from the EUC pattern library is to find the associated EUI pattern based on the UI and SR inserted to find the associated security requirements and IoT technologies. In Step 3, an EUC model then generated using EUC pattern library. In this stage, EUC model also generated by extraction of IoT security requirement from SecReq pattern library and IoT Technologies from IoTTech pattern library label as SubEUC model. The SecREq pattern in the library (derived from SecIoTA model) was associated with EUC model and the collection of textual requirements. Meanwhile, the IoTTech pattern library (derived from SecIoTA model) was associated with SubEUC model. The attributes from IoT technologies are assigned by keyword from EUC Model and textual requirements. In Step 4, an EUI prototype model is the derived from EUC model using EUI pattern library. Here, the associated from user interaction (UI) and system responsibility (SR) from EUC model will derived EUI prototype model form EUI pattern library to generate workable prototype. In Step 5, a workable prototype will be generated to visualize the security requirements of IoT application. In Step 6, the user and requirements engineer as well as the client stakeholders can visualise the security requirements in a form of workable prototype model of a targeted IoT app to be developed. We develop SecIoT_MEReq to realise this approach to elicit security requirements and Iot technologies that must be used by user to develop more secure IoT application.

4.1 EUC and EUI Pattern Libraries

We developed a set of EUI patterns in EUI Pattern Library from a collection of such patterns previously identified by [9] together with the patterns that were developed by [18], which are all applicable across various domains.

Table 1: Example of EUC and EUI Pattern Libraries

EUC Pattern		**EUI Pattern**	**EUI Pattern Category**	**Related Security Requirement**
User Interaction (UI)	**System Responsibility (SR)**			
Identify self Request identity Request identification Access system		Input ID	Input	Authentication Access control
Request status Select status Provide status		Input status		Authorization Availability
Select option Enter option		Input option		Integrity Confidentiality
Enter detail Input detail		Input detail		Authentication
Enter location Select location		Submit location		Integrity
	Verify user Verify identity Verify ID	Display ID	Display	Authorization
	Update status Update option	Display status		Availability Confidentiality

	Display status Display detail	Receive status Receive detail		Access control Availability
	Show information Show location	Display information		Availability
	Show item Display item	View item		Availability
	Validate payment Show payment Show code	Display payment		Integrity Access control Availability
Choose payment Select amount Choose transaction		List payment	List	Integrity Access control
Choose option Choose item Choose event		List option		Confidentiality Availability
	Offer choice Offer solution	List of choice		Availability

4.2 SecReq Pattern Library

We developed a security requirements pattern library (SecReq) to support the elicitation of the security related properties from the related security requirements. The purpose of security requirements pattern library is to help RE to increase correctness issues, especially to elicit security requirements before proceed for application development. The generic nature of the security pattern library contributes to its usefulness as it can be utilized and applied in any domain of IoT application. Further, the usage of pattern library can minimize human time effort for eliciting correct security requirements.

Table 2: Example of SecReq Pattern Library

Example Keyword from EUI Model	Examples Attributes for Security Requirements	Related Security Requirements	Examples of IoT Domain Related
Input ID Enter detail Register Input code and ID	Username Password PIN ID card Fingerprint Retinal pattern Biometric identifier	Authentication	Smart City Domain (Smart Parking System, Warehouse Management, Smart Farming) Health Well-being Domain (Healthcare Monitoring, Children Protection, Home Security) Industrial Domain (Mobile Ticketing, Smart Shopping, Animal Tracking)
Verify ID Check details Provide information	Permission Verify Gain access	Authorization	Smart City Domain (Smart Parking System, Warehouse Management, Smart Farming) Health Well-being Domain (Healthcare Monitoring, Children Protection, Home Security)

			Industrial Domain (Mobile Ticketing, Smart Shopping, Animal Tracking)
List options View detail Display status Show coordinate Provide code	Accessible Obtainable Software patching	Availability	Smart City Domain (Smart Parking System, Warehouse Management, Smart Farming) Health Well-being Domain (Healthcare Monitoring, Children Protection, Home Security) Industrial Domain (Mobile Ticketing, Smart Shopping, Animal Tracking)
Sends data Submit location Request information Choose options Get information Detect signal Process information	Limits access Unreadable data Restricted access	Confidentiality	Smart City Domain (Smart Parking System, Warehouse Management, Smart Farming) Health Well-being Domain (Healthcare Monitoring, Children Protection, Home Security) Industrial Domain (Mobile Ticketing, Smart Shopping, Animal Tracking)
Submit billing Validate data Notify location Confirm transaction Process information Record details	Protect data Unmodified data Unaltered data	Integrity	Smart City Domain (Smart Parking System, Warehouse Management, Smart Farming) Health Well-being Domain (Healthcare Monitoring, Children Protection, Home Security) Industrial Domain (Mobile Ticketing, Smart Shopping, Animal Tracking)
Receive information Transmit data Select amount Generate code	Limited access Control the access	Access control	Smart City Domain (Smart Parking System, Warehouse Management, Smart Farming) Health Well-being Domain (Healthcare Monitoring, Children Protection, Home Security) Industrial Domain (Mobile Ticketing, Smart Shopping, Animal Tracking)

4.3 IoTTech Pattern Library

We developed a IoT technologies pattern library (IoTTech) to support the elicitation of the IoT technologies from the related technologies used in IoT applications. The purpose of IoTTech pattern library is to help RE to increase correctness issues, especially to elicit IoT technologies before proceed for application development.

Table 3: Example of IoTTech Pattern Library

Example Keyword from EUI Model	Example Attributes / Devices of IoT technologies	IoT Technologies	Examples of IoT Domain Related
Check status Detect motion Detect vibration	Temperature Vibration Motion Current detection	Sensor	Health Well-being Domain (Children Protection, Home Security) Industrial Domain (Animal Tracking)
Provide location Display location Make payment Send code Provide notification Search location	North coordinate East coordinate Altitude Signals Locator Identifier Tracker Mobility Connection density Spectral efficiency Latency Peak data rate	Mobile networks (GPS, QR Code, 4G/5G)	Smart City Domain (Smart Parking System, Home Security, Smart Farming)) Health Well-Being Domain (Healthcare Monitoring, Children Protection, Home Security) Industrial Domain (Mobile Ticketing, Smart Shopping,
Identify code Verify code Verify item Check status Scan tags	RFID tags/transponder RFID readers	RFID system	Smart City Domain (Smart Parking System, Warehouse Management) Health Well-Being Domain (Children Protection) Industrial Domain (Smart Shopping, Animal Tracking)
Detect signal Check signal Verify item	Access point Scalability Diversity Hotspot	Wi-Fi (WLAN, IEEE 802)	Smart City Domain (Smart Farming, Smart Parking System) Health Well-Being Domain (Healthcare Monitoring, Children Protection, Home Security) Industrial Domain (Smart Farming)
Detect signal Check signal Confirm status	Packet-based Access point-centered	Bluetooth	Health Well-Being Domain (Healthcare Monitoring, Home Security) Industrial Domain (Mobile Ticketing, Animal Tracking)

Based on Table 2 and Table 3, SecReq and IoTTech pattern library were develop using keyword matching, The system will identify the keyword inserted by user in the textual requirement for various of IoT

application scenario that collected from published literature and verified by experts. When using SecIoT_MEReq, a requirement engineer key in the requirements in the form of user story in the textual requirements text area. Then, EUC models are extracted from textual requirements. This is done by using EUC pattern library. We then map the EUC model into EUI model uisng EUC pattern library. Then the security requiremens and iot technologies are generated and sugeested from the support of SecReq and IoTTech pattern library.

5 EXAMPLE OF USAGE

Daniyal, a requirements engineer would like to elicit the security requirements provided by the client-stakeholder using SecIoT_MEReq. He sits with Sofea, who is the IoT Developer to elicit the requirements, which she had captured earlier. First, he need to login to access to SecIoT_MEReq. Besides, he also can clicks the Home tab which give an information about SecIoT_MEReq. Also, the Definition tab helps them to understand the terms for IoT Domain, Security Requirements and IoT Technologies. To use the tool, he then clicks the Tool tab and the choose IoT domain and the application that listed from the tool. From there, he inserted the textual requirements in the form of business scenario and clicks "Update Model" (Figure 4).

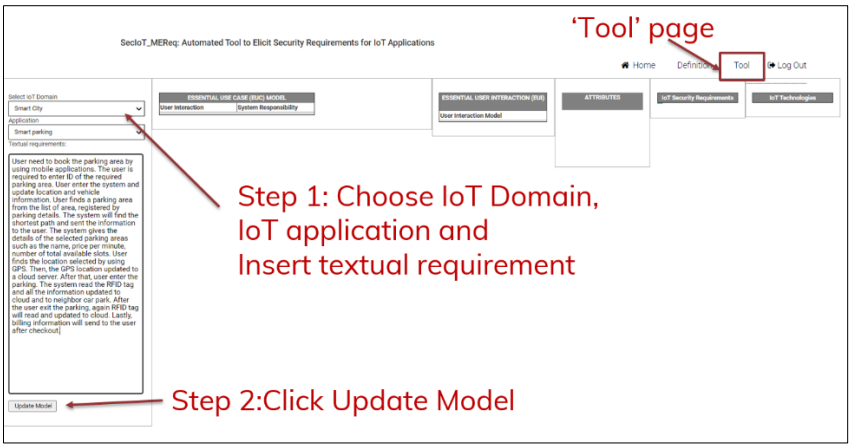

Figure 4: Tool page for SecIoT_MEReq

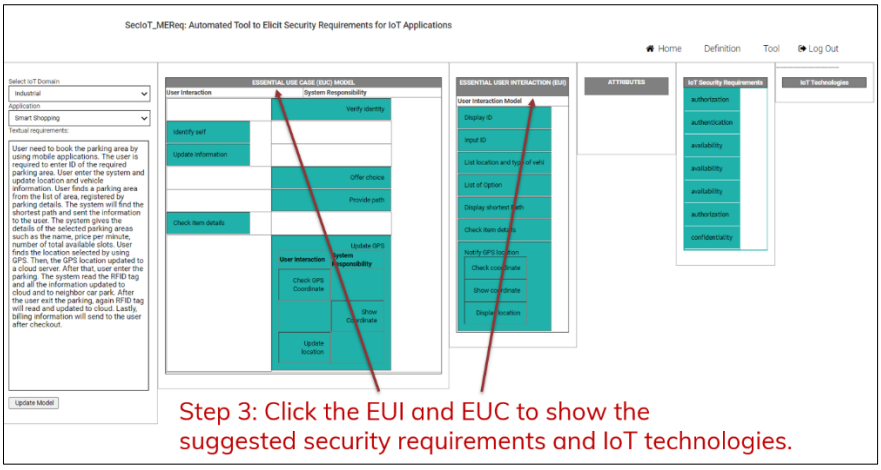

Figure 5: EUI and EUC generated for tool

As show in Figure 5, after Sofea click button, "Update Model", SecIoT_MEReq will generate EUC and EUI from EUC pattern library and EUI pattern library. Here, she then uses the tool to elicit the attributes for the particular EUC and EUI model that can be viewed. As shown in Figure 6, Sofea click on the EUC of "Identify Self" to its user interaction. In this example the EUI model for the EUC model of "Identify Self" is "Input ID". For this case, the attributes display are "username", "password", "PIN", "fingerprint", and "retinal pattern" for suitable security requirement "Authentication". The tool help them to identify the security requirement that they needed to develop an IoT application based on the requirements attributes.

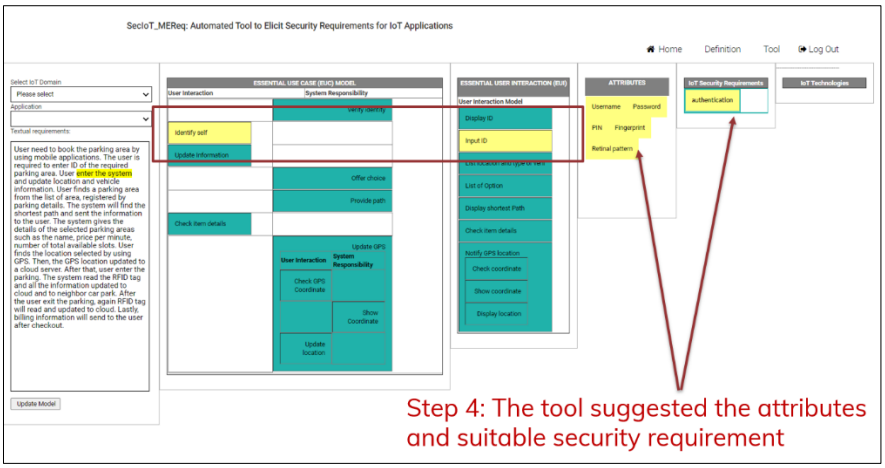

Figure 6: Security Requirements Elicitation in SecIoT_MEReq (1)

As shown in Figure 7, to get better understanding, Sofea that click another user interaction for EUC model "Check item detail" and "Check item detail" also display for EUI model. Here, for this case, tool generate "permission", "verify", and "gain access" for attributes for security requirement "authorization".

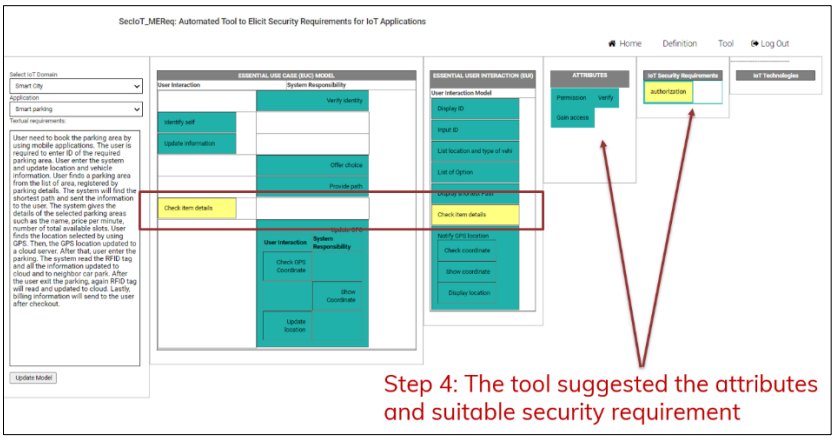

Figure 7: Security Requirements Elicitation in SecIoT_MEReq (2)

Next, as shown in Figure 8, Sofea clicks user interaction "Check GPS coordinate" from update GPS for EUC model and "Check coordinate" for EUI model. Attributes that generated from tool are "fingerprint", "PIN", "password", "username", "retinal pattern". Security requirement suggested by tool is "authentication" and "GPS" for IoT technologies.

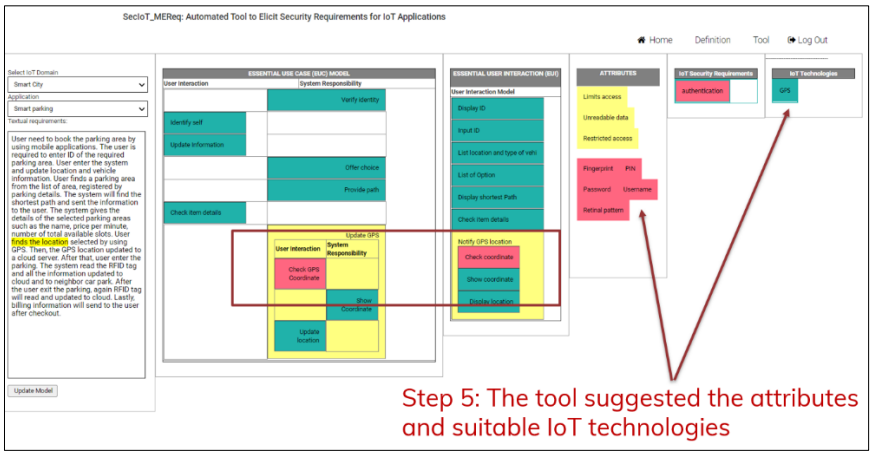

Figure 8: IoT Technologies Elicitation in SecIoT_MEReq (1)

Then, Sofea decide to click system responsibility “Show coordinate” from update GPS for EUC model and “Show coordinate” for EUI model. Attributes that generated from tool are “permission”, “gain access”, and “veirfy”. Security requirement suggested by tool is “authorization” and “GPS” for IoT technologies. She also can click any of the EUC model to generate the security requirement and IoT technologies suggested by tool for the IoT domain.

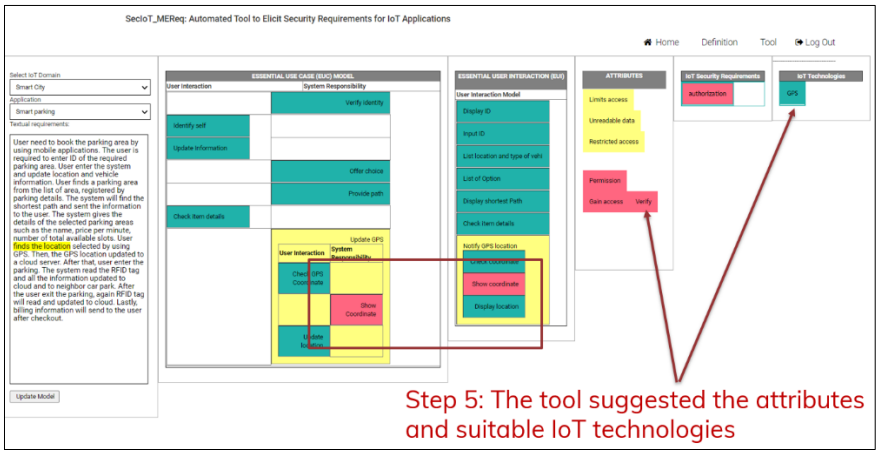

Figure 9: IoT Technologies Elicitation in SecIoT_MEReq (2)

6 ARCHITECTURE AND IMPLEMENTATION

We have developed a prototype tool, called SecIoT_MEReq based on the approach in the previous section. The objective is to assist requirements engineers in eliciting security requirements during requirements elicitation with client-stakeholders. Our tool provides the (1) extraction of EUC and EUI model (2) extraction of security requirement from SecReq; and (3) extraction IoT technologies from IoTTech pattern library. SecIoT_MEreq has been developed using PHP programming language and adopts Model-View-Controller (MVC) design pattern and three-tier architecture. MVC design pattern was implemented to develop a platform-independence software application that supports different platforms, such as mobile devices, tablets, and different browsers on different operating system. As shown in, Figure 10, MVC pattern divides an interactive application into three components: *Model*, *View* and *Controller*.

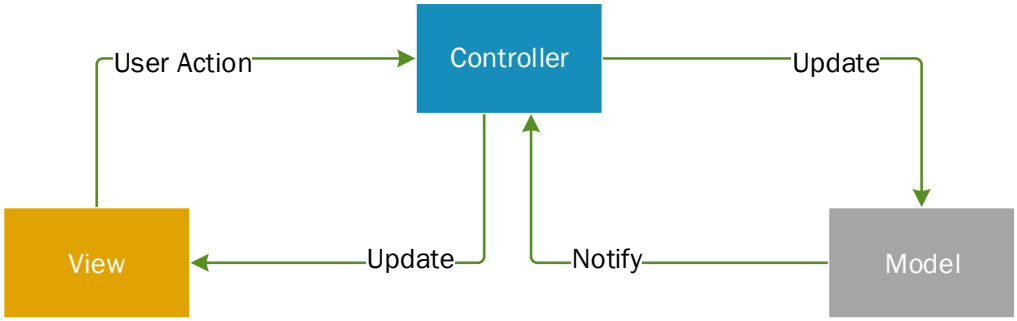

Figure 10: The MVC Design Pattern

The *Model* manages the application's data and business logic, the *View* is in charge of presenting the data or model to the user via the browser, and the *Controller* manages the communication between the model and the view. SecIoT MEReq's development was inspired by the works of [19] [20], and the identification of the associated security elements are based on the definitions from the basic security services.

Figure 11 illustrates the high-level architecture of the SecIoT_MEReq tool that comprises three tiers; presentation, application processing, and data management layer. The layout is three-tier architecture, where each layer is separated from each other. This independency allows for better performance, easier maintenance and more scalable architecture [21].

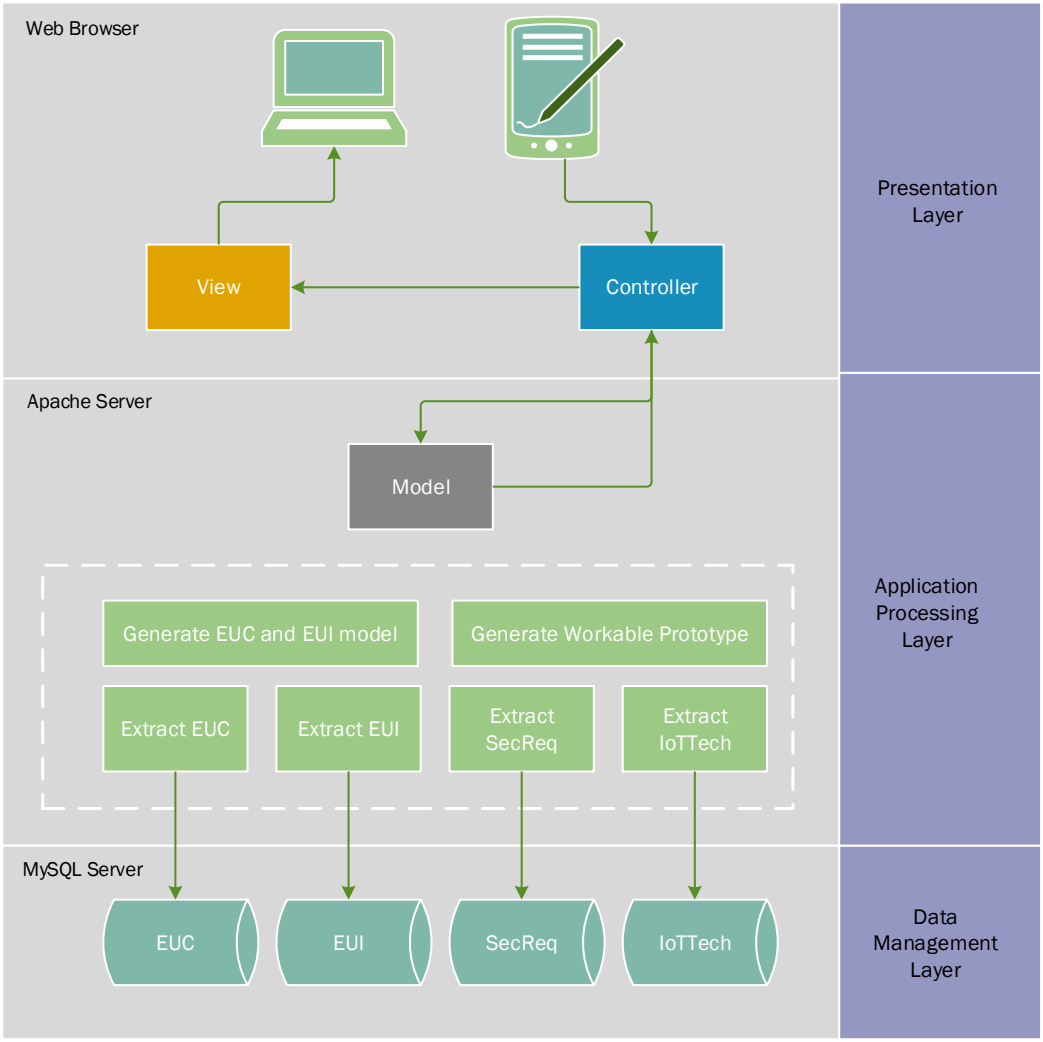

Figure 11: SecIoT_MEReq High Level Architecture

The presentation layer handles the interaction between the users and the system. The *View* and *Controller* exist in the presentation layer. Here, a web client from any platform such as an iPad, mobile phone or desktop can request to access the SecIoT_MEReq tool. The user interacts with the SecIoT_MEReq tool through the *Controller* component. The *Controller* that contains the client-side scripting, handles the http request processing and business logic of the tool. It receives user input as events and translates them into service request for the *Model* or the *View*. When a user accesses the SecIoT_MEReq, the scripts in the *Controller* will determine the type of browser and device used by the user. Then, it will request the correct view from the *View* component. Each view has associated controller component. Next, the *View* component will make requests from the *Model* to fetch the data from business and data layer and display the information to the user.

At the business processing layer, the Apache server hosted the PHP implementation for the main event handlers of SecIoT_MEReq. This contains the key elements for the extraction of security requirements and IoT technologies components from the textual requirements, extraction of application scenario components at the SecIoT_MEReq template editor, analysis and evaluation of the model-based component and business scenario syntax from pattern library.

At the data management layer, MySQL database server contains the EUC, EUI, security requirements and IoT technologies libraries. A Java module supports the extraction of the security requirements and IoT technologies from the generated EUC and EUI models. This process extract both the security requirements and IoT technologies from the SecReq and IoTTech Pattern Libraries. A sequence of SecReq and IoTTech are extracted and visualised together with the workable prototype.

7 EVALUATION AND RESULTS

7.1 Usability Study

We ran two usability tests to assess the usability of our approach and tool for eliciting security requirements for the correctness test. In the first usability test, respondents were required to use our tool and complete a series of questionnaires in order to provide feedback on the usability of our tool. A total of 56 undergraduate students participated in this usability test. The participants are all majoring in Software Engineering and taking the Software Requirement course. During the second usability test, respondents were required to use our tool and respond to semi-structured interview questions about its usability. The interview questions are described in detail. Semi-structured interviews with 12 industrial experts were conducted via Google Meet for this usability test. The purpose of these interviews was to gather their thoughts and perceptions on the usability of SecIoT MEReq, which aids in eliciting security requirements from an IoT industrial standpoint.

This usability study required the participants to perform two main tasks: (1) to explore the tool to accomplish the required tasks and (2) to complete a survey questionnaire upon completion of the tasks. The survey questionnaire was designed to elicit the users' perception regarding the usefulness, ease of use, ease of learning, and satisfaction of the tool based on a five-level Likert scale. The questionnaire also include the Cognitive Dimension (CD) and open-ended questions. We considered Cronbach's Alpha test to measure the reliability of our questionnaire. The alpha coefficient for this group is 0.872, suggesting that the items have relatively high internal consistency. It is proven that the questionnaire has high reliability. This result is based on [22] who claims that a reliability coefficient of 0.70 or higher is considered "acceptable" in most social science research situations.

We conducted the evaluation of our SecIoT_MEReq tool with two group (Group A and Group B) which is focusing on the automatic generation of security requirements and key-textual structures with completeness checking from our two pattern libraries, SecReq and IoTTech. In this evaluation, the participants were requested to perform two tasks:

Evaluation : Elicitation of Security Requirement for IoT Application using SecIoT_MEReq Tool. The tasks for the evaluation are:

a) Task 1 : Exploring the SecIoT_MEReq tool capabilities to automatically generate the analysis of security requirements and IoT technologies from our pattern library, SecReq and IoTTech. Here, the participant were explained with the step using the tool.
b) Task 2 : Demonstration using video demonstration and the observation of the functionality of the tool. Here, the participants are requested to insert the IoT domain, application and textual requirements into the tool to generate the suggestion of attributes, security requirements and IoT technologies.

7.1.1 Video Demonstration and Observation Result for Task 1 and Task 2

In this evaluation, the participants communicated through video conferencing using Zoom application, with the researcher/observer to gain understanding of using the tool. Overall, we found that most of the participants were interested using the tool as it is able to automatically generate the security requirements and IoT technologies from our pattern libraries, EUC, EUI, SecReq and IoTTech. The tool also helps them to see the analysis for textual requirement by highlighting the related components. However, we got a few feedbacks from the participants indicated that they were uncomfortable with the the font size display in the tool where the font is to small that make them difficult to read the content. These comments were noted for future work.

Participants were asked about their proficiency in using the SecIoT_MEReq tool and their experience in using any tools similar to our SecIoT_MEReq, before they moved to usability and CD notation study. The results are shown in Figure 12 and Figure 13.

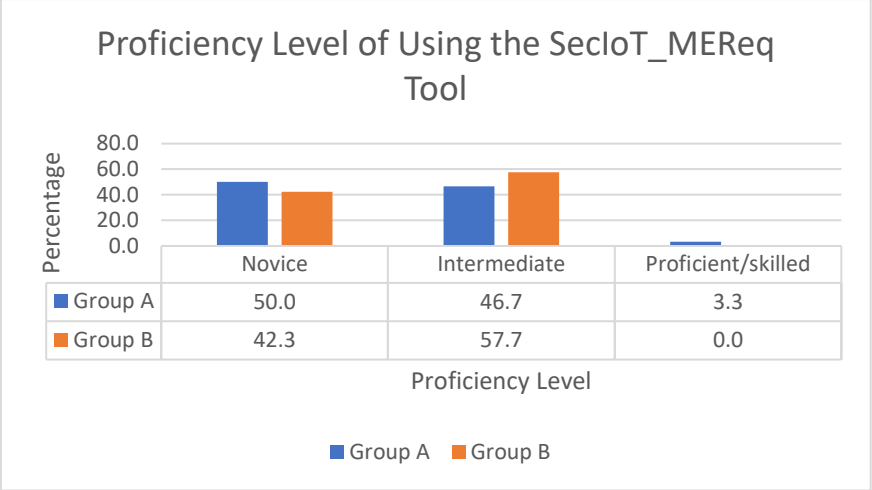

Figure 12: Proficiency level of Using the SecIoT_MEReq Tool

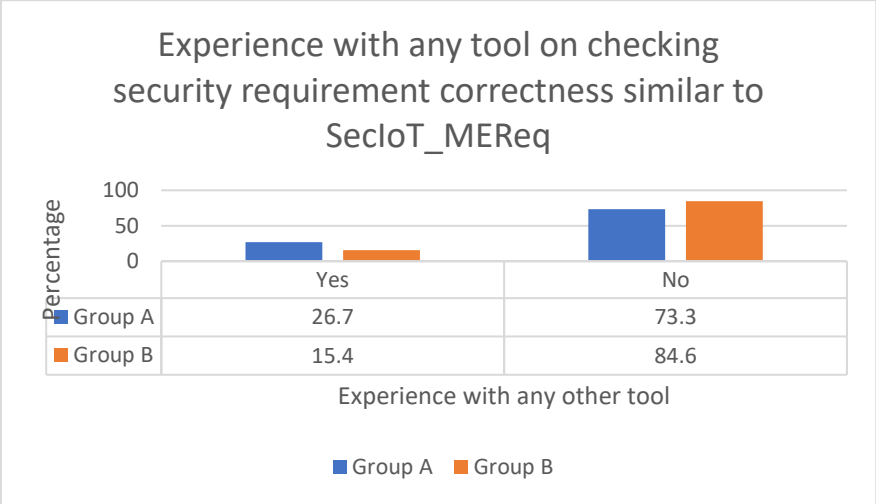

Figure 13: Experience with any tool on checking security requirement correctness similar to SecIoT_MEReq

Based on the background results provided, from Group A and Group B, all of the participants were identified as novice and intermediate in using the SecIoT_MEReq tool and the web design tool used to construct SecIoT_MEReq. This indicates that they have some background in using web-based tool design. Most of them were thus unfamiliar with tool design. Less than 30% of the participant have experience in using familiar tool. But, based on the result, we concluded that these groups of users were unfamiliar with RE tools like SecIoT_MEReq. The same group of participants was also used in our next phase of evaluation. The results of the usability criteria and CD study based on the questionnaire are shown in Figure 14, Figure 15, Figure 16, Figure 17, Table 4 and Table 5. Figure 14 presents the results of each usability criterion. For each criterion, the results of each corresponding three-question block were averaged to produce the results shown. There is a strong agreement among the participants in terms of the usefulness of the tool, where 90% (Group A) and 98.7% (Group B) of the participants strongly agreed or agreed that the tool was useful to assist them in eliciting security requirements because it automate the process and makes the eliciting process of security requirements easier. We have also found that 87.8% (Group A) and 94.9% (Group B) of the participants felt that the tool was easy to use because the interface design of the tool is user-friendly and they found less inconsistencies in the tool. In terms of ease of learning, 88.9% (Group A) and 97.4% (Group B) of the participants claimed that it was very easy to learn since the flow and the interface design of tool is simple and user-friendly. Additionally, 80% (Group A) and 97.5% (Group B) of the participants were satisfied with the tool as there were no special technical skills required to write complete security requirements. There were only a small number of participants form both group (less 10%) who were undecided or disagree on the usefulness aspect of the tool. For Ease of Use, there was a slight disagreement

(2.2% from Group A and 1.3% from Group B) where the participants feel uncomfortable with certain terms/keywords while dealing with the security requirements components. Overall, the usability results show that our prototype tool is useful, easy to use and learn. Users also expressed their high level of satisfaction when using the tool.

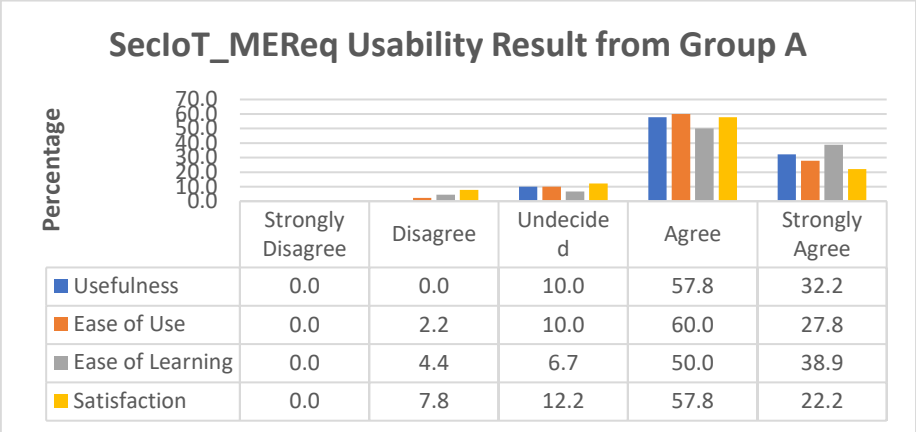

Figure 14: Usability Study of SecIoT_MEReq from Group A

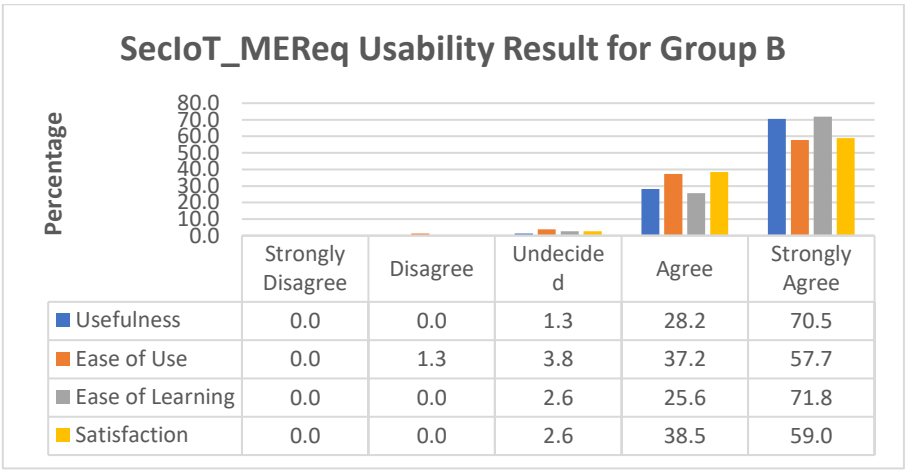

Figure 15: Usability Study of SecIoT_MEReq from Group B

Figure 16, Figure 17, Table 4 and Table 5 shows the result of the CD study. The CD study allows us to explore in more detail the reasons for users' perceptions as well as further discussion on the strengths and weaknesses of the tool. The results are based on percentage depending on the number of participants' answers for each scale.

Table 4: CD Study Result of SecIoT_MEReq from Group A

Cognitive Dimension	Strongly Disagree (%)	Disagree (%)	Undecided (%)	Agree (%)	Strongly Agree (%)
Visibility	0.00	0.00	13.33	**70.00**	**16.67**
Viscosity	0.00	0.00	13.33	**66.67**	**20.00**
Diffuseness	0.00	0.00	23.33	**56.67**	**20.00**
Hard Mental Effort	**10.00**	**53.33**	10.00	20.00	6.67
Error-proneness	**10.00**	**40.00**	20.00	26.67	3.33
Closeness of mapping	0.00	0.00	16.67	**66.67**	**16.67**
Consistency	0.00	0.00	13.33	**70.00**	**16.67**
Hidden dependencies	0.00	0.00	10.00	**80.00**	**10.00**

Cognitive Dimension	Strongly Disagree (%)	Disagree (%)	Undecided (%)	Agree (%)	Strongly Agree (%)
Progressive Evaluation	0.00	0.00	16.67	**66.67**	**16.67**
Premature Commitment	0.00	0.00	16.67	**60.00**	**23.33**

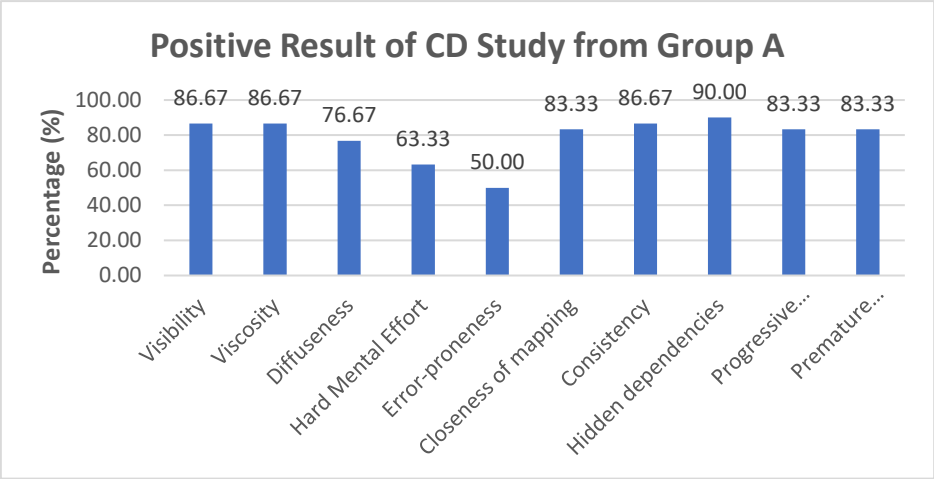

Figure 16: Positive Result of CD Study of SecIoT_MEReq from Group A

Table 5: CD Study Result of SecIoT_MEReq from Group B

Cognitive Dimension	Strongly Disagree (%)	Disagree (%)	Undecided (%)	Agree (%)	Strongly Agree (%)
Visibility	0.00	7.69	7.69	**38.46**	**46.15**
Viscosity	0.00	3.85	11.54	**46.15**	**42.31**
Diffuseness	0.00	3.85	0.00	**61.54**	**34.62**
Hard Mental Effort	**3.85**	**65.38**	11.54	11.54	7.69
Error-proneness	**3.85**	**57.69**	11.54	7.69	19.23
Closeness of mapping	0.00	0.00	3.85	**57.69**	**38.46**
Consistency	0.00	0.00	3.85	**57.69**	**38.46**
Hidden dependencies	0.00	3.85	3.85	**73.08**	**19.23**
Progressive Evaluation	0.00	0.00	7.69	**50.00**	**42.31**
Premature Commitment	0.00	0.00	15.38	**50.00**	**34.62**

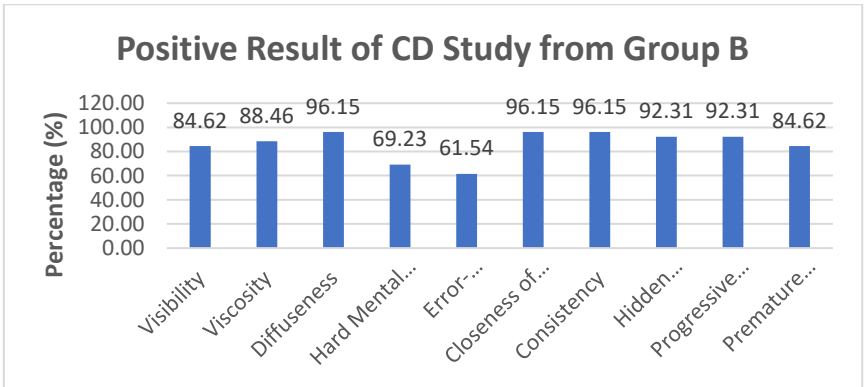

Figure 17: Positive Result of CD Study of SecIoT_MEReq from Group B

We summarize the results for each dimension as follows.

Visibility : About 86.67% (Group A) and 84.62% (Group B) of the participants either strongly agreed or agreed that they can see the various part of the tool that show clearly the five components requirements: textual natural language requirements in the textual editor, EUC model, EUI model, test requirements and test cases. They could also easily see the dependencies of each component as a visual link and highlights are provided. The remaining 13.33% (Group A) and 7.69% (Group B) of the participants are undecided and 7.69% participants from Group B are in disagreement.

Viscosity: 86.67% (Group A) and 88.46% (Group B) of the participants either strongly agreed or agreed that the tool allowed them to make changes easily to the textual requirements. Strong result of both visibility and viscosity show that the participants from both group were comfortable with the tool.

Diffuseness : 76.67% (Group A) and 96.15% (Group B) of the participants were strongly agree or agree that the notation used by the tool is succinct and not long-winded and 13.33% of the participants from Group A were undecided. However, 3.85% of the participants from Group B were disagree and thought it was hard to understand the notation when using it for the first time. They were confused the different between security requirements and the attributes.

Hard Mental Effort: About 63.33% (Group A) and 69.23% (Group B) of the participants either strongly disagreed or disagreed that this tool needs a lot of effort to solve the tasks. They were quite contented as this tool is able to extract the security requirement automatically, which minimises a lot of their time and effort. However, there was still some dissatisfaction from 26.67% (Group A) and 19.23% (Group B) of the participants who thought this tool still required effort to understand the notation and the layout when using it for the first time.

Error-proneness: 50% (Group A) and 61.54% (Group B) of the participants either strongly disagreed or disagreed that the tool leads the user to make errors. This is because the extracted security requirement is believed to be accurate as all the security attributes and IoT technologies patterns are already pre-defined in the library. However, 21.54% of the participants from both Group A and B were undecided: they may have believed that the tool could be constrained by the size of the library.

Close of Mapping: Most participants (83.33% from Group A and 96.15% from Group B) either strongly agreed or agreed that the notation used was closely related to the results. They understood the labels used to describe the requirement components. Only 10% of the participants from Group A and 3.85% from Group B were undecided with the notation used.

Consistency: 86.67% (Gorup A) and 96.15% (Group B) of the participants were either strongly agreed or agreed that they could easily identify the requirements components: textual natural language, EUC model, EUI model, test requirements and test cases throughout the task. Only 13.33% (Group A) and 3.85% (Group B) of the participants were undecided: they were unsure about the different between security requirements and security attributes but believed that the notations used were consistent and straight-forward.

Hidden Dependencies: 90% (Group A) and 92.31% (Group B) of the participants either strongly agreed or agreed that the dependencies among the three requirements components were visible. Visual links are provided to show the dependencies between all requirements components (textual requirements, EUC model, EUI model, test requirements and test cases) when trace-back is performed. Highlights with yellow color help to visualise the dependencies among components. Only 10% of the participants from Group A and 3.85% from Group B were undecided with this.

Progressive Evaluation: 83.33% (Group A) and 92.31% (Group B) of the participants also either strongly agreed or agreed that SecIoT_MEReq allows users to evaluate their work at any time and to verify the security requirement produced by the library. Here, participants could make changes to the textual requirements if they did not agree with the tool‘s decision. Only 16.67 % and 7.69% from both Group A and B were undecided with this dimension.

Premature Commitment : This dimension reflects the sequence of using this tool in order to achieve the results. 83.33% (Group A) and 84.62% (Group B) of the participants strongly agreed or agreed that the tool allows a user to perform the task from any direction. Another 16.67% (Group A) and 15.38% (Group B) were undecided.

The questionnaire also contains open-ended questions to obtain the participants' feedbacks about the positive points of the tool. Based on our analysis of the keywords provided by the participants, there were two main themes identified: 'useful' and 'ease of use'. Table 6 shows the result of the open-ended question related to the positive points of the tool given by the participants.

Table 6: Frequency Table for the Result of Open-ended Question

	Group A		Group B	
Keywords	**Frequency**	**Percentage (%)**	**Frequency**	**Percentage (%)**
Useful	11	36.67	12	46.15
Ease of Use	9	30	11	42.31

Based on Table 7, almost 82.82% (Group A and B) of the participants felt the tool was useful to assist them in eliciting complete security requirements because the tool automate the process of eliciting security requirements. Among the similar responses stated from Group A were "*It is good as it can help me to elicit the requirement correctly*", "*The function works well and can help a lot in elicit the requirements*" and "*The tool is efficient and it does not consume too much time comparing to eliciting manually*". Another response from group B were stated as "*The tool is very straightforward and very helpful in eliciting the security requirements*", "*Help to elicit more quickly*" and "*Because SecloT_MEReq can help user to protect data*". These responses show that the tool was useful and helpful for them to automatically elicit the correct security requirements from the pattern libraries.

Table 7: Frequency Table for the Result of Open-ended Question

Keywords	**Group A**		**Group B**	
	Frequency	**Percentage (%)**	**Frequency**	**Percentage (%)**
Nothing to be improved	25	83.33	21	87.77
Better interface design	4	13.33	3	11.54
Provide user manual	-	-	2	7.69
Others	1	1.33	-	-

Based on Table 7, the top suggestion is nothing to be improved, the top suggestion is nothing to improved, which received about 83.33% from Group A and 87.77% from Group B. Among the similar comments were "*none*". Another 13.33% from Group A and 11.54% from Group B suggest for better interface design. Among the similar comments by the participants were "*the UI can be upgraded from time to time*", "*make user interface more interesting*", "*improve the interface*" and "*improve the interface of the system*". Another 7.69% from Group B suggest to provide user manual based on the suggestion "*make user manual*" and "*put user manual to ease the user*". Only 1.33% participant from Group A give feedback to make "*Mobile version*". In summary, the results from this usability study showed a positive feedback from the participants. This indicates that our tool is useful and easy to use to assist in eliciting security requirements. The ultimately can help to decrease the development cost.

Figure 18 shows the result of comparison of positive result of CD study for Group A and Group B. Based on the result, the were a similarity result for Group A and B. The result shows that both group have a same thought of the CD study for the SecIoT_MEReq for visibility (Group A – 86.67%, Group B – 84.62%), viscosity (Group A – 86.67%, Group B – 88.46%), hard mental effort (Group A – 63.33%, Group B – 69.23%), error-proneness (Group A – 50.00%, Group B – 61.54%), closeness of mapping (Group A – 83.33%, Group B – 96.15%), consistency (Group A – 86.67%, Group B – 96.15%), hidden dependencies (Group A – 90.00%, Group B – 92.31%), progressive evaluation (Group A – 83.33%, Group B – 92.31%) and premature commitment (Group A – 83.33%, Group B – 84.62%). Both of the group agreed that our tool have high positive result of cognitive dimension. While the result for diffuseness have a little bit differences (Group A – 76.67%, Group B – 96.15%) because of the confusion between security requirements and attributes. It was hard to them to understand the notification because they were using it for the first time.

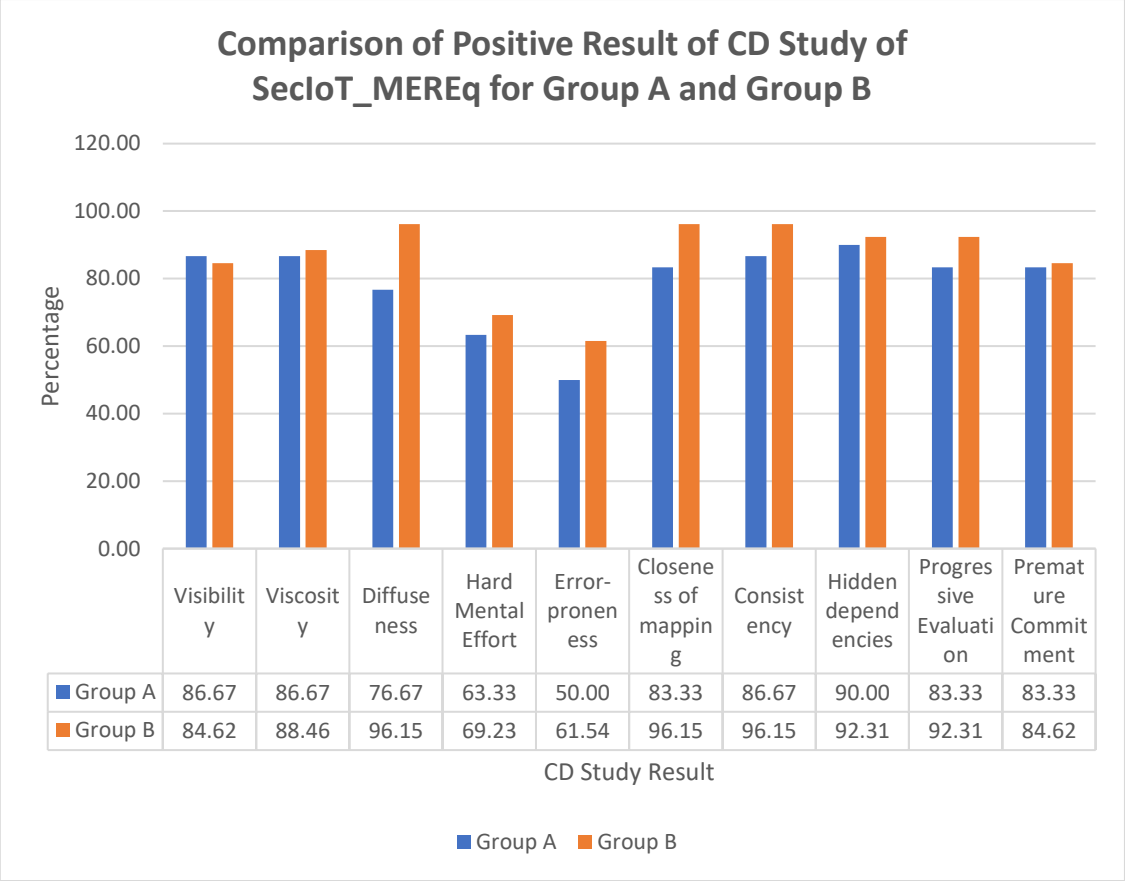

Figure 18: Comparison of Positive Result of CD Study of SecIoT_MEReq from Group A and Group B

As shown in Figure 19, the result of the opinion of useful and ease of use of SecIoT_MEReq for Group A and Group B have a similarity. Both of the student agreed that our tool, SecIoT_MEReq is useful (Group A – 55%, Group B – 52.17%) and ease of use (Group A – 45%, Group B – 47.12%). They agreed that our tool was useful to help and assist the user in eliciting complete security requirements because the tool automate the process of eliciting security requirements. They also felt that out tool can ultimately can help to decrease the development cost.

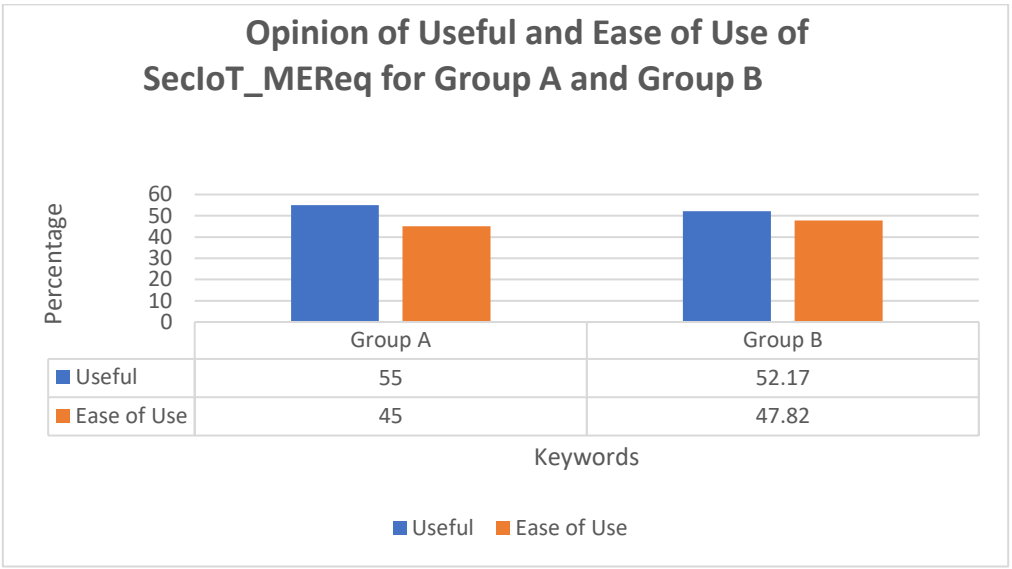

Figure 19: Opinion of Useful and Ease of Use of SecIoT_MEReq from Group A and Group B

7.2 Use of SecIoT_MEReq by requirements engineering professionals

The second usability test was conducted to gain the feedbacks from the experts in the field of software development and testing regarding the usability of SecIoT_MEReq. This usability test was conducted with 12 industrial experts where all participant's opinion regarding SecIoT_MEReq is analysed. Semi-structured interviews were carried out and demonstration of SecIoT_MEReq was conducted using Google Meet application. The demographic of participants are specifically in the field of software engineer, system analyst, programmer, IT analyst and IT application. They have between two and fifteen years working experience in the IoT industry. Prior to the interview, we informed them the purpose of the interview and defined the different terminologies and definitions used in our interview questions to ensure the consistency of responses. We provided a brief description of our tool and show them the video demonstration and gave them the access to a link that provides them the overview the tool using samples of textual requirements from their recent projects.

Based on the interview results and comments, it indicates that they agreed that SecIoT_MEReq tool is simpler and easier to be understood and learnt. It also helps to reduce the time and effort in eliciting security requirements of IoT applications. We concluded that our automated tool approach is useful and helpful to requirements engineer to elicit security requirements for Internet of Things (IoT) applications based on their feedback that indicates the words "helpful, very helpful, really helps" etc. Most of the participants also agreed that our tool can help them to visualize elicitation security attributes and then conduct early elicitation of the security requirements of IoT application. This result is similar with the finding that we have found from the open-ended questions in our survey. From the interview, all of the experts agreed that our SecIoT_MEREeq tool is useful and helpful in eliciting correct security requirements. They found that the auto-elicitation of the requirements really helps for better understanding for security requirements correctness. They agreed that this tool automates the eliciting process, provides guidance on how to elicit security requirements. This tool provides early elicited and helps to reduce mistakes at the early stage. These could avoid repeatable tasks and reduce cost in eliciting security requirements. Besides, it will avoid the increase of timeframe because of required task to redo by the system analyst and IoT developer. This tool also helps for elicit IoT technologies suitable used for develop secure IoT application that will lead to good/quality application. All of the experts also expressed their satisfaction that this tool could help in assisting the process of eliciting security requirements. All of them have never used any tools that are similar to SecIoT_MEREeq.

Based on the two usability tests conducted, we concluded that our automated tool approach is useful and helpful to requirements engineers to elicit security requirements. Most of the participants also agreed that our tool can help them to elicit the correctness of security requirements and IoT technologies before proceeding to the development stage.

8 CONCLUSION

Requirement engineers need to elicit security requirements for IoT application at an early stage of development. We have developed an automated tool called SecIoT_MEReq for security requirements of mobile apps by employs the idea of Model-Driven Development (MDD) using semi formalised EUCs and EUIs prototype model. Evaluation of our prototype tool with real security examples and end users shows positive results. However, we faced several limitations, but they can be ameliorated in future works. First, the size of the security requirement (SecReq) and IoT technologies (IoTTech) library. This research focuses on developing security requirement pattern library drawn from a collection of literature review and requirements from industries. Hence, it is limited to the functional security requirements. In this case, eliciting non-functional requirements is beyond the scope of this approach. Secondly, our tool can only extract the prototypes that were predefined in our libraries. Here, the libraries were designed and developed based on our case studies and sample requirements/scenarios collected from various sources of IoT domain. Therefore, a wider collection of keywords was required to enhance the scalability of the library. We believe, this research arises from the need to have an automated approach for eliciting security requirements and correctness checking to support requirements engineers and client-stakeholders in the process to ensure the correctness of the security requirements named SecIoT_MEReq. This tool is still at the prototype stage. Although this approach has a strong potential in assisting the requirements engineer community and IoT developers in eliciting security requirements, there are some future works that need to be done. We hope to extend the tool capabilities by embedding AI technique for the keyword searching to allow the

tool to be trained to automatically extract the newly found components for other domains in our libraries. And also, the enhancement of the visualization view and layout of SecIoT_MEReq tool with embedded user manual for novice user or beginner user. This improvement is helpful in guiding the novice users to use the tool easily.

ACKNOWLEDGEMENT

The authors would like to gratefully acknowledge the assistance and funding made available by the Universiti Teknikal Malaysia Melaka (UTeM) and the Ministry of Higher Education, Malaysia under the Fundamental Research Grant Scheme (FRGS), grant no.: FRGS/1/2022/ICT03/UTEM/01/1. Also, thanks to Politeknik Sultan Azlan Shah (PSAS) for its support and all those who participate in the study and helped to facilitate the research process.

REFERENCES

[1] E. Alsaadi and A. Tubaishat, "Internet of Things : Features, Challenges, and Vulnerabilities," *International Journal of Advanced Computer Science and Information Technology*, vol. 4, no. 1, pp. 1–13, 2015.

[2] B. Russell, C. Garlati, and D. Lingenfelter, "Security Guidance for Early Adopters of the Internet of Things (IoT)," *Mobile Working Group Peer Reviewed Document*, no. April, 2015.

[3] K. Rose, S. Eldridge, and L. Chapin, "The Internet of Things: An Overview Undertanding the Issues of a More Connected World," 2015.

[4] C. Atkinson and T. Kühne, "Model-Driven Development: A Metamodeling Foundation," *IEEE Software*, vol. 20, no. 5, pp. 36–41, 2003, doi: 10.1109/MS.2003.1231149.

[5] S. Winkler and J. von Pilgrim, "A Survey of Traceability in Requirements Engineering and Model-Driven Development," *Software and Systems Modeling*, vol. 9, no. 4, pp. 529–565, 2010, doi: 10.1007/s10270-009-0145-0.

[6] J. David and G. Whittam, "Model-Driven Development," *IEEE Software*, 2008.

[7] R. France and B. Rumpe, "Model-Driven Development of Complex Software: A Research Roadmap," *FoSE 2007: Future of Software Engineering*, pp. 37–54, 2007, doi: 10.1109/FOSE.2007.14.

[8] R. Biddle, J. Noble, and E. Tempero, "From Essential Use Cases to Objects," in *forUSE 2002 Proceedings*, 2002, vol. 1, no. 978.

[9] 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.

[10] S. W. Ambler, "Essential (Low Fidelity) User Interface Prototypes," 2003.

[11] R.-M. Abraham, P. J. Escamilla-Ambrosio, J. Happa, and E. Ahuirre-Anaya, "GARMDROID : IoT Potential Security Threats Analysis Through the Inference of Android Applications Hardware Features Requirements," *Applications for Future Internet*, vol. 2, pp. 63–74, 2017, doi: 10.1007/978-3-319-49622-1.

[12] P. K. Dhillon and S. Kalra, "A Lightweight Biometrics Based Remote User Authentication Scheme for IoT Services," *Journal of Information Security and Applications*, vol. 0, pp. 1–16, 2017, doi: 10.1016/j.jisa.2017.01.003.

[13] H. A. B. Taiseera, P. R. F. Sampaio, and P. Laucopoulos, "Eliciting and Prioritizing Quality Requirements Supported by Ontologies: A Case Study using ElicitO Framework and Tool," *Expert System*, vol. 00, no. 00, 2012, doi: 10.1111/j.1468-0394.2012.00625.x.

[14] S. Hilde, H. Shareeful, and K. Schneider, "Eliciting Security Requirements and Tracing Them to Design: An Integration of Common Criteria, Heuristics, and UMLsec," *Security Requirements Engineering*, vol. 15, no. Special Issues, pp. 63–93, 2010, doi: 10.1007/s00766-009-0093-9.

[15] P. Gope and T. Hwang, "BSN-Care: A Secure IoT-Based Modern Healthcare using Body Sensor Network," *IEEE Sensors Journal*, vol. 16, no. 5, pp. 1368–1376, 2016.

[16] J. Wurm, K. Hoang, O. Arias, A. R. Sadeghi, and Y. Jin, "Security analysis on consumer and industrial IoT devices," *Proceedings of the Asia and South Pacific Design Automation Conference, ASP-DAC*, vol. 25, pp. 519–524, 2016, doi: 10.1109/ASPDAC.2016.7428064.

[17] G. Loniewski, E. Insfran, and S. Abrahão, "A Systematic Review of The Use of Requirements Engineering Techniques in Model-Driven Development," *Lecture Notes in Computer Science*, vol. 6395 LNCS, no. PART 2, pp. 213–227, 2010, doi: 10.1007/978-3-642-16129-2_16.

[18] M. Kamalrudin, J. Hosking, and J. Grundy, "MaramaAIC: Tool Support for Consistency Management and Validation of Requirements," *Automated Software Engineering*, pp. 1–45, 2016. doi: 10.1007/s10515-016-0192-z.

[19] M. Kamalrudin, J. Grundy, and J. Hosking, "Tool Support For Essential Use Cases To Better Capture Software Requirements," in *ACM The International Conference on Automated Software Engineering, ASE 2010*, 2010, pp. 255–264. doi: 10.1145/1858996.1859047.

[20] M. Kamalrudin, J. Grundy, and J. Hosking, "Automated Support for Consistency Management and Validation of Requirements," The University of Auckland, 2011.

[21] Ian Sommerville, *Software Engineering, 7th Edition*. 2004.

[22] G. UCLA: Statistical Consulting, "What does Cronbach's alpha mean.docx." 2016.