Third Asia Pacific Conference on Advanced Research (APCAR, Melbourne, July, 2016)
ISBN:978 0 9943656 20
www.apiar.org.au

CAPTURING SECURITY REQUIREMENTS OF MOBILE APPS USING MobiMEReq

Noorrezam Yusop[a], Massila Kamalrudin[b], Safiah Sidek[c]
[abc] UniversitiTeknikal Malaysia Melaka, Melaka, Malaysia
Corresponding email: p031320001@student.utem.edu.my

Abstract

Mobile devices have been widely used around the world as it facilitates interaction between people and services at anywhere and anytime. Mobile applications are found to be used for conducting online transactions, saving data and exchanging information. However, issues related to security have become a major concern among mobile users as insecure applications may lead to security vulnerabilities that make them to be easily compromised by hackers. Further, mobile applications that lack of security concerns also could exposed the mobile devices and users to malwares that could cause failure and malfunction to the application. Thus, it is important for mobile apps developers to capture security requirements of mobile apps at the earliest stage to prevent security problems during the implementation of mobile application. In this paper, we describe our automated approach and tool support, called MobiMEReq that helps to automatically capture the security attributes requirements of mobile apps. We represent our extraction using Essential Use Cases (EUCs) and Essential User Interface (EUI) prototype models. We also compared our tool with the manual approach to evaluate the performance and correctness of our tool in various application domains. The results of the study show that our tool is able to help requirements engineers to easily capture security-related requirements of mobile apps.

Keywords: Security Requirements, Security Attributes, Mobile Apps, Extraction Security Related Requirements.

1.Introduction

Mobile devices are widely used as they allow interaction between people and services anywhere and anytime. The use of mobile app is rapidly growing especially in online transactions, such as online purchasing, flight booking and hotel booking. This has led to a plethora of applications being developed to fulfil the needs of mobile users. However, many mobile app developers tend to ignore the security aspect of the application at the early stage of development, leading to malicious attacks and security breaches. It is also found that most of the time, engineers fail to capture correct security related requirements during the elicitation phase as they faced difficulty to understand the terms and knowledge of the security (Schneider et al., 2011). Further, the process of capturing correct and consistent requirements from client-stakeholders is often difficult, time consuming and error prone (Kamalrudin&Grundy, 2011; Paja et al., 2012). Therefore, there is a need for automation support to capture security related requirements at the early stage of mobile apps development.

In our previous work (Yusop et al., 2015), we have conducted a user study to gauge requirements engineers' ability to capture the security related requirements from a set of business requirements of a mobile app. The study found that almost 60% incorrect security attributes were captured by the participants for each of the requirements given. This shows that requirements' engineers face difficulties to capture the security related requirements, especially when extracting the security attributes (Yahya et al., 2014; Yusop et al., 2015). Further, it was found that the longest time taken by the participants to extract the security attributes is more than 45 minutes, implying that more effort is needed to perform this task. These findings have motivated us to 1) develop an automated tool support for capturing and extracting security requirements and to 2) evaluate the tool to demonstrate its ability to enhance the accuracy and usability for capturing security requirements of mobile apps.

This paper describes the development of a tool support that is able to automatically capture security requirements of mobile apps using Essential Use Cases (EUCs) and Essential User Interface (EUI) prototype models. We begin by describing the background of EUCs and EUI models. We then present our prototype tool for the extraction of security attributes of the security requirements. Next, we describe an experiment that compares its performance in extracting security attributes to the same requirements samples as per discussed in (Yusop et al., 2015). We also discuss an experiment that aims to prove its accuracy in extracting a range of security attributes from various sets of requirements. We also present the users' perception on its usability. Finally, we discuss the implications of these studies and the prototype, as well as our future work.

2.Background

2.1 Security requirements attributes

Security requirements are classified as non-Functional Requirements (NFRs) and they are related to system confidentiality, integrity and availability. Security requirements attribute or security attribute can be defined as any piece of information that may be associated with a controlled implicit entity or user for the purpose of implementing a security policy that is not necessarily implemented directly in data structures (Ivo et al., 2009). The most common security related requirements for software are shown in Figure 1 labeled as A (Ian et al., 2006), while the attributes used for each security related requirements are shown in Figure 1 labeled as B.

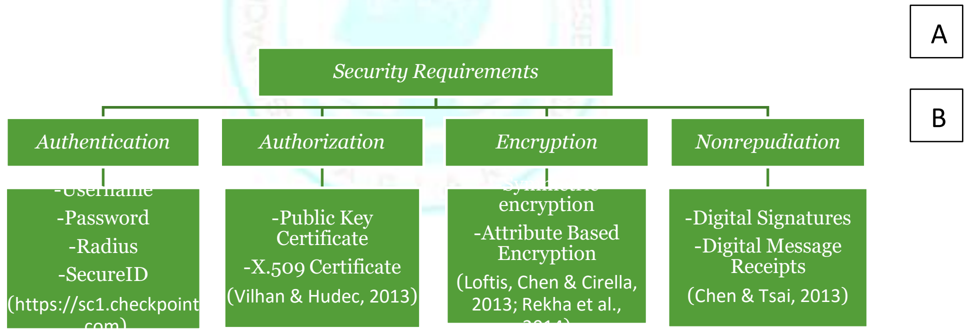


Figure 1: Security requirements and its related security attributes

The terms for all of these security requirements and security attributes are specifically found for software or system only. Based on our previous work (Yusop, Kamalrudin&Sidek, 2015), the security requirements are found similar for mobile application development and they are normally considered at the later phase of the system or mobile application development.

2.2 Essential Use Cases (EUCs)

The EUC approach has been defined by Constantine and Lockwood 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*" (Constantine&Lockwood, 1999). Its main objectives are to support better communication between the developers and stakeholders via a technology-free model and to assist better requirements capture. These objectives can be achieved by allowing only specific details relevant to the intended design to be captured (Biddle, Noble&Tempero, 2002). Figure 2 shows an example of natural language requirements (lefthand side) and an example of EUC (right hand side) when capturing the requirements (adapted from Constantine and Lockwood, 1999). The natural language requirements from which the important phrases are extracted (highlighted in yellow) are shown on the left hand side of Figure1.

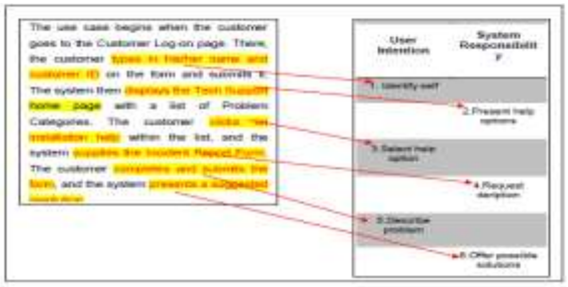


Figure2: Example of Textual Natural Language Requirements (left) and Example of Essential Use Case (EUC model) (right)

When capturing requirements from natural language text, the EUC approach is found to be more suitable than the conventional UML use case. An equivalent EUC description is generally shorter and simpler than a conventional UML use case as it only comprises the essential steps (core requirements) of user's intrinsic interest. Specifically, 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. The abstractions are mainly used for specifying the steps of the use case rather than narrating the use case as a whole. It has also been shown (Yahya et al., 2014) that EUCs are beneficial for capturing security requirements. They have developed a security pattern library comprising Security related EUC, called the SecEUCs and security related essential interaction termed as the SecEI. Examples of the SecEI and the SecEUC areas shown in Table 1.

2.3 Essential Use Interface (EUI)

EUI prototyping is a low fidelity prototyping approach (Ambler, 2003-2009). 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 (Constantine&Lockwood, 2003). EUI prototyping extends from and works in tandem with the semi-formal representation of EUCs that also focuses on the users and their usage of the system, rather than the system features (Ambler, 2004). It thus helps to avoid clients and REs from being misled or confused by chaotic, evolving and distracting details. EUI also allows some explorations on the usability aspects of a system. Figure 3 shows examples of EUI prototypes developed from EUC models.

Table 1: Example of SecEUC Pattern Libraries (Yahya et al., 2014)

SecEI	**SecEUC**
Check username	Identify self
Check password	
Verify username	
Make payment	Make payment
Complete payment form	

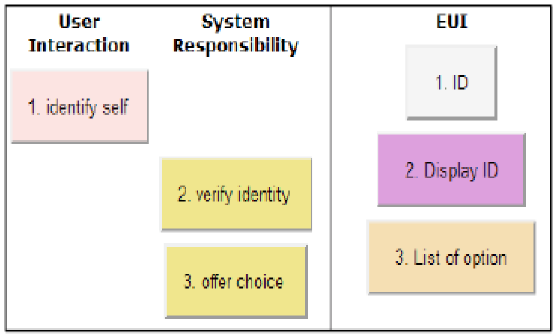


Figure 3: Examples of EUI prototype from EUC models

2.4 Related work

Currently, for security guidance and solution, most developers or engineers refer to the Common Criteria (CC), although the CC is a bit complex and difficult to understand by novice (Paja et al., 2012). Further, developers tend to make mistakes when determining the right security requirements and attributes. This is necessary to identify the requirements and attributes personally without any supports, such as the automation or the manual training. Therefore, CC presents the requirements in two distinct categories:1) the functional requirements, and 2) the assurance requirements. In security behavior, the CCis described in both types (Ware&Bowles, 2005). Moreover, there is no predefined instruction provided to the user when using the GUI for dynamic analysis. This leads to various challenges in completing the security identification process (Aho, Menz&Raty, 2011; Kull, 2012). These instances justify the need for an automation that can help, especially the novice to elicit security requirements and attributes. To tackle the issues mentioned, Haley et al. (2008) has proposed an approach to support security requirements elicitation and analysis. They applied a method to construct a system context using a problem-oriented notation. However, due to the complexity of the proposed approach, experts to construct the setting and analysis are needed. Berger, Sohr and Koschke (2013)also conformed to the claim that software engineers lack the security knowledge although this body of knowledge is easily accessible. The software engineers and developers face difficulties to extract and make decision on selecting the relevant piece of security knowledge to be applied to their design or requirements.

3. Our approach

Motivated from the problems and challenges found in our previous work (Yusop et al., 2015), we have developed an approach and automated tool called MobiMEReq to assist requirements engineers to automatically capture an extraction of the security-related requirements for mobile apps.

3.1 Mobile Sec Attributes Pattern Libraries

We developed a *Mobile SecAttribute Pattern Library* that consists of SecEUC, SecEUI and related security attributes for mobile apps. Both the SecEUC and SecEUI were derived from a collection of security attribute requirements of mobile apps from real security requirements taken from the industry and published material (Yusop, Kamalrudin&Sidek, 2015). The collection of SecEUC patterns from the EUC and EUI models was based on the methodology defined by Yahya et al. (2014). The selection of the security attributes were based on our analysis (Yusop, Kamalrudin&Sidek, 2015)and a survey conducted with the industry regarding the most important security attributes for mobile apps development. Our Mobile Sec Attributes Pattern library also consists of defined test requirements and test cases. We also recognise that there is a direct relationship between the SecAttributes and the SecEUC.

As shown in Figure 4, the SecEUC can be associated with one or many security attributes. For example, 'Identify self', an abstract interaction of a SecEUC, is associated with three security attributes, which are the 'username', 'password' and "biometric id". This kind of relationship exists because the security attributes has direct association with the security related requirement of authentication and authorization.

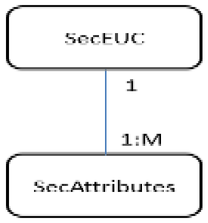


Figure 4: The relationship between SecEUCmodel,

Table 1: Sample of our SecAttributes Pattern

SecEUC	**SecAttribute**	**SecCtrl**
Make payment	Username	Authentication Transaction
	Password	
	PaymentID	
Verify TAC Code	Username	Authentication Authorization Transaction
	Password	
	TAC Code ID	
	PaymentID	

3.2 Using our approach

We have developed a prototype tool that supports the capture process of security requirements for mobile apps. This prototype tool is an extension of our earlier (Kamalrudin, Grundy&Hosking, 2010) tool that runs in both mobile and web applications. This tool assists security requirements engineers to capture an initial security attribute extracted from textual natural language requirements and the SecEUC. Both the requirements engineer and client-stakeholder can capture the security attributes for the security related requirements of mobile app using MobiMEReq itself or mobile apps to MobiMEReq tool.

Figure 5 shows an overview of our approach and the role of the SecAttribute Pattern Library when extracting the security requirements of mobile apps at the early stage of requirements extraction. By implementing the SecEUC and SecEUI model work by Yahyaet al. (2014), our approach is able to derive Capture of Security attributes from a set of mobile app requirements.
Capture of Security Attributes: Here, the Mobile SecAttribute Pattern Library is used by the tracing engine to analyse the textual mobile requirements and to map the matching set of abstract interactions of SecEUC (A1). This approach allows Requirements Engineers (REs) to capture the important security attributes from the textual mobile requirements gathered from client-stakeholder (1). Then, the textual requirements are mapped to SecEUC and SecEUI model (2). As shown in Figure 5, the SecAttribute is generated to visualize the security attributes that best fit to the generated SecEUC and SecEUI model based on the defined attributes in the

Mobile SecAttribute Pattern Library (3). Next, the requirements engineer (RE) can visualise the security requirements as a form of workable rapid prototype (4) model of the targeted mobile app.

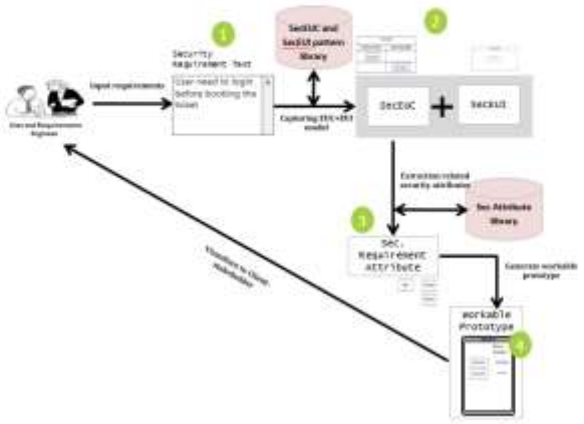

1
Security Requirement Text
User need to login before booking the ticket
Input requirements
SecEUC and SecEUI pattern library
Capturing EUC+EUI model
2
SecEUC
SecEUI
Extraction related security attributes
Sec Attribute library
3
Sec. Requirement attribute
Generate workable prototype
Workable Prototype
4

Figure5: An overview of our proposed approach.

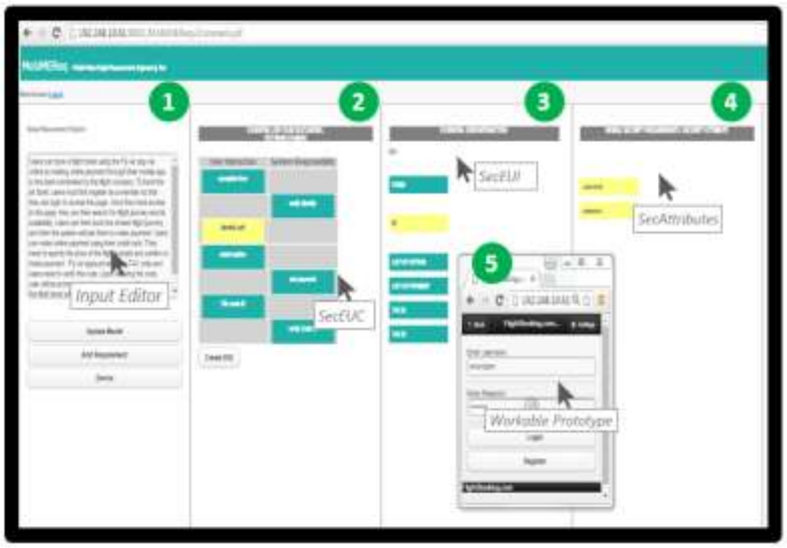


Figure 6: Example of tool usage for integration security attributes and visualization tool, MobiMEReq.

3.3 Tool Support

We have developed a prototype tool for the automated extraction of security requirements for mobile apps. Figure 6 shows the components of the prototype tool such textual requirement, EUC, EUI, Security attribute and workable prototype. The component focuses on the extraction as well as the captured mobile security requirements, which are captured from client stakeholders. This component focuses on capturing the mobile security requirements which begins with the textual mobile security requirements. In relation to this, the textual requirements based on user scenario are written in the textual editor (1). Further, the tool generates the SecEUC models that show the user intention and system responsibility (2) and SecEUI models that show the essential user interface (3). The visualised security attributes are generated (4) and the user can test the data from the workable prototype [A](5).

4. Evaluations and Results

We have conducted two studies to evaluate the effectiveness of our new automated tool for mobile apps security requirements engineering. First, we evaluated the accuracy of the captured security attributes. Next, we conducted a usability study with 50 undergraduate students.

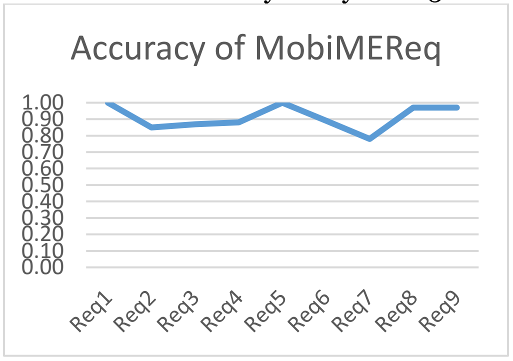


Figure 7: Accuracy across different set of security requirements

4.1 Accuracy

We have evaluated the accuracy of our tool by applying nine use case scenarios from nine different applications of domains derived from different researchers, developers and ourselves. The types of applications are Medical, Nutrition Controller, Car park booking, Hostel booking, Car rental booking, etc. The accuracy derived from the tool was evaluated by comparing the answers with the actual interaction pattern developed by us following Constantine and Lockwood's methodology, as well as Yahya et al. (2014).

The evaluation results shown in Figure 7 describe the accuracy of our tool applied in different requirements. This shows that there is variability across a range of requirements. The average correctness of the tool is approximately 91%. However, we failed to get 100% correctness due to the formation of the EUC and EUI models and the linguistic issues, which was also found by Kamalrudin, Grundy and Hosking (2010). The incomplete generated EUC and EUI models exist because they depend on the correct extraction from natural language or pattern editor.

User study: Manual vs. Automatic Extraction

In order to further evaluate the efficacy of the tool, we then compared the accuracy and performance of the tool with manual requirements extraction by 50 novice from our previous study (Yusop et al., 2015). The novice users in this context are the undergraduate students who have the limited background of software engineering and requirements engineering, as well as EUC and EUI models. As shown from Table 3, our tool is able to capture more accurate security attributes than the manual approach, with 95% correctness in comparison to 46% correctness from the manual approach reported in (Yusop, Kamalrudin&Sidek, 2015). The tool failed to capture only one security attribute (amount) for login requirements. The incorrectness is due to the passive structure phrases written in the requirement, which are not defined in the library. This result shows that MobiMEReq is able to facilitatethe participants to capture accurate security attribute. Further, the automated extraction process takes just over 1 second to execute in comparison to the average time (30 minutes) taken by the manual approach to extract the security attributes by the participant in (Yusop, Kamalrudin&Sidek, 2015).

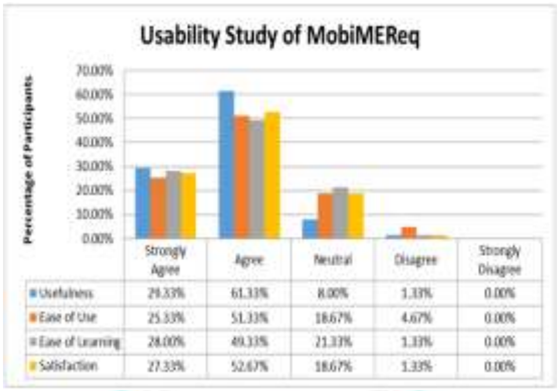
Table 2: Comparison result of correctness between manual extraction and automated validating tool.

Functional Requirement	Answers	No. Correct answers		No. Wrong answers	
		Manual	Tool Tracing	Manual	Tool Tracing
Register	Username	42	1	8	0
	Password	40	1	10	0
	Passport/Ic	33	1	17	0
	BookingId	30	1	20	0
Login	Amount	48	0	2	1
	TACCode	47	1	3	0
Search Flight	FlightNo	17	1	33	0
	Destination	9	1	41	0
	Username	8	1	42	0
	Password	7	1	43	0
Book Flight Ticket	BookingCode	9	1	41	0
	FlightNo	14	1	36	0
	Username	11	1	39	0
	Password	10	1	40	0
Payment Ticket	Account No	43	1	7	0
	TAC Code	47	1	3	0
	TicketID	6	1	44	0
	Username	9	1	41	0
	Password	7	1	43	0
Correctness ratio		437	18	513	1
		46%	95%	54%	5%

5.Usability study

We then conducted a survey to investigate their perceptions and the experience of users using the tool. The survey was conducted with the same participants in the manual study. The survey questionnaire focuses on the participant's evaluation of the tool with respect to its usefulness, ease of use, ease of learning and satisfaction. In addition, participants were also requested to write their comments on the four aspects mentioned earlier. The results of this survey are shown in Figure 7. As shown in Figure 7, we found that 90.66% of the participants felt that the tool is useful for generating and extracting the list of security attributes, while only 1.33% of the participants disagreed and 8% of them were neutral. However, they suggested that the tool could be more useful if it is embedded within a tool that visually displays the EUCs. They highlighted that such visualizations would allow them to better understand the interaction between the textual requirements, the EUC and EUI model to generate related security attributes. 76.66% of the participants agreed that the tool is easy to use, while only 4.67% disagreed that the tool is easy to use. 18.67% percent of the respondents were indifference or neutral. Further, they commented that they like and are interested in using the tool. However, they would like to have a better user interface for security attributes with a more user-friendly design rather than just the preliminary prototype.

With respect to ease of learning, 77.33% of the respondents agreed that the tool is easy to learn. 21.33% of the participants were indifferent, while only 1.33% disagreed with the statement. The participants commented that they would like to have a better tutorial or video to overcome the difficulties of the learning process. With respect to satisfaction, 80% claimed that they were satisfied with the tool. 18.67% of the participants were indifferent and only 1.33% were dissatisfied with the tool. They commented that the tool still needs further enhancement. Based on the results of the survey, overall we can conclude that the tool is useful, user friendly and easy to learn. In general, most of the participants were satisfied with the MobiMEReq tool. The participants also evaluated the response time of the MobiMEReq tool for the validating process. All participants have the opinion that the tool performs faster than the manual approach.



	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
🟦 Usefulness	29.33%	61.33%	8.00%	1.33%	0.00%
🟧 Ease of Use	25.33%	51.33%	18.67%	4.67%	0.00%
⬜ Ease of Learning	28.00%	49.33%	21.33%	1.33%	0.00%
🟨 Satisfaction	27.33%	52.67%	18.67%	1.33%	0.00%

Figure 8: Usability study for capturing security requirements for mobile app.

Conclusion

We have developed an automated capturing approach and tool support called MobiMEReq for security requirements of mobile apps by using EUCs and EUIs prototype model. Our evaluation indicates that our approach and tool support MobiMEReq is able to automatically capture the security attributes of mobile apps.Our future work is to providethe end-to-end validation approach that can capture and validate security requirements for mobile apps. We also plan to embed an intelligent algorithm to prioritise the test data; hence, allowing a random set of data to be used for testing.

Acknowledgement

We would like to thank UniversitiTeknikal Malaysia Melaka for its support and Science fund grant: 01-01-14-SF0106 and also Ministry of Education (MOE), MyBrain15.

References

i. Aho, P., Menz, N. &Raty, T., 2011. *Enhancing Generated Java GUI Models with Valid Test Data*. Proceeding of the 2011 IEEE Conf. on Open Systems (ICOS),Langkawi, Malaysia, pp. 310-315.

ii. Ambler, S. W., 2004. *The Object Primer: Agile Model-Driven Development with UML 2.0*, 3rd edn. New York: Cambridge University Press.

iii. Berger, B. J., Sohr, K. &Koschke, R., 2013. *Extracting and Analyzing the Implemented Security Architecture of Business Applications*. Proceeding of the 17th European Conference on Software Maintenance and Reengineering.

iv. Biddle,R., Noble, J. and Tempero, E. 2002. Essential Use Cases and Responsibility in Object Oriented Development. *Proceeding of the 25th Australasian Computer Science Conference, Australian Computer Society, Inc. Chicago*, 24(1), pp. 7-16.

v. Chen, C. L. &Tsai, W. C., 2013. Using a Stored-Value Card to Provide an Added-Value Service of Payment Protocol in VANET. Proceedings of theInnovative Mobile and Internet Services in Ubiquitous Computing (IMIS), Seventh International Conference, Taichung, pp. 660 -665.

vi. Constantine, L. L. & Lockwood, A. D. L., 2003. *Usage-Centered Software Engineering: An Agile Approach to Integrating Users, User Interfaces, and Usability into Software Engineering Practice*. Proceeding of the 25th International Conference on Software Engineering (ICSE'03) 2003, IEEE Computer Society, Portland, Oregon.

vii. Constantine, L. L. & Lockwood, L. A., 1999. *Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design*. Pearson Education.

viii. Haley, C. B., Laney, R. C., Moffett, J. D. &Nuseibeh, B., 2008. *Security Requirements Engineering: A Framework for Representation and Analysis*. IEEE Trans. Software Eng, pp.133-153.

ix. Ian, G., 2006. *Essential Software Architecture*. pp. 1-283.

x. Ivo, F. K., George, E., Leslie, C., Leana, G. &Nenad, M., 2009. *A Comprehensive Exploration of Challenges in Architecture-Based Reliability Estimation*. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), pp. 202-227.

xi. Kamalrudin, M. & Grundy, J., 2011. *Generating Essential User Interface Prototypes to Validate Requirements*. Proceedings of the 2011 26th IEEE/ACM International Conference on Automated Software Engineering, pp. 564-567.

xii. Kamalrudin, M., Grundy, J. & Hosking, J., 2010. *Managing Consistency between Textual Requirements*. Abstract Interactions and Essential Use Cases. Proceeding of the 2010 IEEE 34th Annual Computer Software and Applications Conference, pp. 327–336.

xiii. Kull, A., 2012. *Automatic GUI Model Generation: State of the Art*. Proceeding of the 2012 IEEE 23rd Int. Symposium on Software Reliability Engineering Workshops (ISSREW),Dallas, TX, USA, pp. 207-212.

xiv. Loftis, C. E., Chen, T. X. &Cirella, J. M., 2013. Attribute-Level Encryption of Data in Public Android Databases. (RTI Press publication OP-0016-1309). Research Triangle Park, NC: RTI Press.

xv. Paja, E., Dalpiaz, F., Poggianella, M. & Roberti, P., 2012. *STS-tool: Socio-Technical Security Requirements through Social Commitments*. Proceeding of the Conference 21st IEEE International Requirements Engineering Conference (RE), pp. 331-332.

xvi. Rekha, A., Anitha, P., Subaira, A. S. &Vinothini, C., 2014. A Survey on Encryption Algorithms for Data Security. *IJRET: International Journal of Research in Engineering and Technology*, 3(12), pp. 131-134.

xvii. Schneider, K., Knauss, E., Houmb, S., Islam, S. &Jurjens, J., 2011. Enhancing Security Requirements Engineering by Organizational Learning. *Requirements Engineering*, 17(1),pp. 35-36.

xviii. Vilhan, P. &Hudec, L., 2013. *Building Public Key Infrastructure for MANET with Help of B.A.T.M.A.N. Advanced*. Modelling Symposium (EMS), European, Manchester, pp. 566 -571.

xix. Ware, M. S. &Bowles, J. B., 2005. *Using the Common Criteria to Elicit Security Requirements with Use Cases*. Proceedings of the IEEE, pp. 273-278.

xx. Yahya, S., Kamalrudin, M., Safiah, S. & Grundy, J., 2014. *Capturing Security Requirements Using Essential Use Cases (EUCs)*. Proceedings of theFirst Asia Pacific Requirements Engineering Symposium, APRES, Auckland, New Zealand, pp. 16-30.

xxi. Yusop, N., Kamalrudin, M. &Sidek, S., 2015. Security Requirements Validation for Mobile Apps: A Systematic Literature Review. *JurnalTeknologi (Science & Engineering)*, 77(33), pp. 123-137.

xxii. Yusop, N., Kamalrudin, M., Yusof, M. M. &Sidek, S.,2015. *Challenges in Eliciting Security Attributes for Mobile Application Development*. Proceeding of the Conference KSII The 7th International Conference on Internet (ICONI), Kuala Lumpur, Malaysia.

Other Resources:

xxiii. Ambler, S. W., 2003-2009. *Essential (Low Fidelity) User Interface Prototypes*. Available at: www.agilemodeling.com/artifacts/essentialUI.htm

xxiv. User Authentication in Mobile Access. Available at: https://sc1.checkpoint.com/documents/R77/CP_R77_Mobile_Access_WebAdmin/41587.htm [Accessed September 2015]