A QUALITATIVE STUDY ON BEST PRACTICES AND PROCESS OF ELICITING UNAMBIGUOUS QUALITY ATTRIBUTES IN SCRUM-BASED PROJECTS

HUSSIN AHMED ABELKADER

DOCTOR OF PHILOSOPHY
UNIVERSITI UTARA MALAYSIA
2022

Awang Had Salleh
Graduate School
of Arts And Sciences

Universiti Utara Malaysia

PERAKUAN KERJA TESIS / DISERTASI
(Certification of thesis / dissertation)

Kami, yang bertandatangan, memperakukan bahawa
(We, the undersigned, certify that)

HUSSIN AHMED ABDELKADER MAHMOUD

calon untuk Ijazah **PhD**
(candidate for the degree of)

telah mengemukakan tesis / disertasi yang bertajuk:
(has presented his/her thesis / dissertation of the following title):

"A QUALITATIVE STUDY ON BEST PRACTICES AND PROCESS OF ELICITING UNAMBIGUOUS QUALITY ATTRIBUTES IN SCRUM-BASED PROJECTS"

seperti yang tercatat di muka surat tajuk dan kulit tesis / disertasi.
(as it appears on the title page and front cover of the thesis / dissertation).

Bahawa tesis/disertasi tersebut boleh diterima dari segi bentuk serta kandungan dan meliputi bidang ilmu dengan memuaskan, sebagaimana yang ditunjukkan oleh calon dalam ujian lisan yang diadakan pada : ***16 Jun 2021.***
That the said thesis/dissertation is acceptable in form and content and displays a satisfactory knowledge of the field of study as demonstrated by the candidate through an oral examination held on:
16 June 2021.

Pengerusi Viva: *(Chairman for VIVA)*	**Assoc. Prof. Ts. Dr. Mohd Hasbullah Omar**	Tandatangan *(Signature)*
Pemeriksa Luar: *(External Examiner)*	**Assoc. Prof. Dr. Rodina Ahmad**	Tandatangan *(Signature)*
Pemeriksa Dalam: *(Internal Examiner)*	**Dr. Mawarny Md Rejab**	Tandatangan *(Signature)*
Nama Penyelia/Penyelia-penyelia: *(Name of Supervisor/Supervisors)*	**Prof. Ts. Dr. Azham Hussain**	Tandatangan *(Signature)*

Tarikh:
(Date) ***16 June 2021***

Permission to Use

In presenting this thesis in fulfilment of the requirements for a postgraduate degree from Universiti Utara Malaysia, I agree that the University Library may make it freely available for inspection. I further agree that permission for the copying of this thesis in any manner, in whole or in part, for scholarly purpose may be granted by my supervisor(s) or, in their absence, by the Dean of Awang Had Salleh Graduate School of Arts and Sciences. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to Universiti Utara Malaysia for any scholarly use which may be made of any material from my thesis.

Requests for permission to copy or to make other use of materials in this thesis, in whole or in part, should be addressed to :

Dean of Awang Had Salleh Graduate School of Arts and Sciences
UUM College of Arts and Sciences
Universiti Utara Malaysia
06010 UUM Sintok

ABSTRAK

Kualiti perisian adalah sangat penting untuk kepuasan pengguna dan kejayaan perisian dalam pasaran. Kajian terkini mendapati beberapa atribut kualiti perisian yang tidak jelas menghasilkan perisian yang rendah kualiti, dan kekurangan amalan elisitasi dalam projek yang menggunakan metodologi Agile Software Development (ASD) terutamanya Scrum. Walau bagaimanapun, dari metodologi ASD semasa, semakan karya bersistematik, dan kajian dari penyelidik terdahulu tidak memberikan penjelasan tentang amalan elisitasi yang diperlukan. Oleh itu, kajian kualitatif ini adalah penting untuk mencapai dua objektif: meneroka amalan terbaik dan mengenal pasti proses untuk mendapatkan atribut kualiti yang jelas dalam projek berasaskan Scrum. Kajian ini menggunakan pendekatan kualitatif di mana data dikumpul melalui temu bual lapan pengamal perisian berpengalaman dari India dan analisis dokumen yang menerangkan dokumentasi atribut kualiti dalam Scrum. Untuk analisis data, kaedah analisis tematik digunakan untuk menganalisis skrip dan dokumen temu bual berkenaan untuk mendapatkan atribut kualiti yang jelas. Dapatan kajian terdiri daripada tiga tema awal yang mewakili tiga langkah dalam proses elisitasi dan enam subtema yang mewakili amalan elisitasi. Langkah pertama ialah pendedahan proaktif kepada atribut kualiti yang terdiri daripada dua amalan: memahami skop perisian dan memikirkan potensi atribut kualiti. Langkah kedua ialah perbincangan pembelajaran bersama yang terdiri daripada dua amalan: meningkatkan pengetahuan teknikal pelanggan dan pengguna dan menyusun perincian atribut yang berkualiti. Langkah ketiga ialah mengesahkan pemahaman umum yang terdiri daripada dua amalan: penggunaan artifak visual dan dokumentasi atribut kualiti. Sumbangan kajian ini memberikan lanjutan kepada badan pengetahuan ASD mengenai keberkesanan penggunaan istilah yang jelas dalam domain perisian, memudahkan istilah teknikal, mewakili artifak perisian boleh guna semula, menunjukkan perisian serupa, membina model kasar dan membangunkan bukti konsep untuk mendapatkan sifat kualiti yang jelas. Tambahan pula, penemuan ini menekankan sumbangan praktikal kepada pembangun perisian seperti mengurangkan usaha, masa dan kos mereka bentuk dan pembinaan perisian.

Kata kunci: Sifat kualiti yang jelas, Kaedah Agile, Scrum, Amalan elisitasi keperluan.

ABSTRACT

Software quality is very crucial for users' satisfaction and software success in the market. Recent studies found some ambiguous software quality attributes that may lead to low-quality software, and lack of elicitation practices in projects that apply Agile Software Development (ASD) methodology especially Scrum. However, current ASD methodologies, systematic literature reviews and surveys did not provide explanation of the necessary elicitation practices. Therefore, this qualitative study was essential to achieve two objectives: exploring the best practices and identifying process of eliciting unambiguous quality attributes in Scrum-based projects. The study used qualitative approach in which data was collected via interviewing eight experienced software practitioners from India and documents analysis that explains documentation of quality attributes in Scrum. For data analysis, the thematic analysis method was used for analysing interviews scripts and documents. The findings resulted in three initial themes that represent three steps in the elicitation process and six sub-themes that represent the elicitation practices. The first step is proactive exposure to quality attributes which consists of two practices: understanding software scope and envisaging potential quality attributes. The second step is mutual learning discussion which consists of two practices: ameliorating technical knowledge of customers and users and compiling details of quality attributes. The third step is verifying common understanding which consists of two practices: utilization of visual artefacts and documentation of quality attributes. The contribution of the study provides an extension to ASD body of knowledge regarding the effectiveness of disambiguation of terminologies in software domain, simplifying technical terms, representing reusable software artefacts, showing similar software, drawing mock-up and developing proof of concept in eliciting unambiguous quality attributes. Furthermore, the findings accentuate practical contributions to the software developers such as reducing effort, time and cost of designing and construction of software.

Keywords: Unambiguous quality attributes, Agile methods, Scrum, Requirements elicitation practices.

ACKNOWLEDGEMENT

In the Name of Allah, the Most Gracious the Most Merciful

First and foremost, all praise is due to Allah the Almighty, the Most Merciful, the Most Compassionate for all sorts of blessings. Peace and blessings of Allah the Almighty be upon all Prophets, the Prophet Muhammad, his Family, his Companions and whosoever follows him. I would like to express my gratitude and appreciation to my father, my mother, my brother and my sisters who have always supported me and have been my pillar of strength.

I would like to express my gratitude and appreciation to my supervisor Prof. Ts. Dr. Azham Hussain for the continuous support of my PhD study and research, for his enthusiasm, patience, motivation, and immense knowledge. His unique support promoted me academically and psychologically to overcome multiple hurdles during my research. My gratitude also goes to Assoc. Prof. Dr. Fauzia Baharom, for her insightful instructions that enlighten my thinking about academic research. My research would not have been possible without their guidance.

I am deeply grateful to my colleagues who supported me during the PhD journey, especially for the motivation, discussions and suggestions on the better ways to perform my study.

TABLE OF CONTENTS

List of Tables

List of Figures

List of Appendices

List of Abbreviations

FSL	Formal Specification Language
INCOSE	International Council on Systems Engineering
QAS	Quality Attributes Scenario
SEI	Software Engineering Institute
SLR	Systematic Literature Review
SMS	Systematic Mapping Study
SWEBOK	Software Engineering Body of Knowledge

CHAPTER ONE
INTRODUCTION

1.1 Introduction

This chapter highlights the main aspects of the study which starts with the background of the study, followed by the discussion on the problem. Then, research questions are elucidated in order to construct the objectives. Finally, this chapter presents research scope, research approach and research significance.

1.2 Background of the Study

The findings of a phenomenological study (Bonuke, 2020), multiple case study (Otiji, 2020), a ground theory (Gibbs, 2015) and a large-scale survey with practitioners from software industry (Fricker, Grau & Zwingli, 2015) underlined that a common understanding of requirements among all stakeholders in software projects is a prerequisite for software success. Requirements define what customers and users need in the software to be developed (Sommerville, 2015; Hull, Jackson, & Jeremy, 2010). While users ordinarily use the software, customers are those persons who request for software development (Nuseibeh & Easterbrook, 2000). Therefore, a common understanding of requirements among customers and users who need the software and practitioners who are responsible for developing the software is indispensable to guarantee software success.

In order to ensure a common understanding of requirements among all stakeholders, requirements must be unambiguous. A requirement is unambiguous if it has only one

REFERENCES

Adams, K. M. (2015). *Nonfunctional requirements in systems analysis and design*. Cham, Switzerland: Springer international publishing.

Al-Saqqa, S., Sawalha, S., & AbdelNabi, H. (2020). Agile software development: Methodologies and trends. *International Journal of Interactive Mobile Technologies*, *14*(11).

Alam, S., Nazir, S., Asim, S., & Amr, D. (2017). Impact and challenges of requirement engineering in agile methodologies: A systematic review. *Int. J. Adv. Comput. Sci. Appl*, *8*(4), 411–420.

Aldave, A., Vara, J. M., Granada, D., & Marcos, E. (2019). Leveraging creativity in requirements elicitation within agile software development: A systematic literature review. *Journal of Systems and Software*, *157*.

Aljallabi, B. M., & Mansour, A. (2015). Enhancement approach for non-functional requirements analysis in Agile environment. *Computing, Control, Networking, Electronics and Embedded Systems Engineering (ICCNEEE), 2015 International Conference*, 428–433.

Ambreen, T., Ikram, N., & Usman, M., & Niazi, M. (2018). Empirical research in requirements engineering: trends and opportunities. *Requirements Engineering*, *23*(1), 63–95.

Amorndettawin, M., & Senivongse, T. (2019). Non-functional requirement patterns for Agile software development. *Proceedings of the 2019 3rd International Conference on Software and E-Business*, 66–74.

Anand, R. V., & Dinakaran, M. (2016). Popular agile methods in software development: Review and analysis. *International Journal of Applied Engineering Research*, *11*(5), 3433–3437.

Anwer, F., Aftab, S., Shah, S. M., & Waheed, U. (2017). Comparative analysis of two popular agile process models: Extreme programming and scrum. *International Journal of Computer Science and Telecommunications*, *8*(2), 1–7.

Appan, R., & Browne, G. J. (2012). The impact of analyst-induced misinformation on the requirements elicitation process. *MIS Quarterly*, 85–106.

Arcos-Medina, G., & Mauricio, D. (2019). Aspects of software quality applied to the process of agile software development: A systematic literature review. *International Journal of System Assurance Engineering and Management*, *10*(5), 867–897.

Avgeriou, P., Grundy, J., Hall, J. G., Lago, P., & Mistrík, I. (2011). *Relating software requirements and architectures*. Springer Science & Business Media.

Bajpai, V., & Gorthi, R. P. (2012). On non-functional requirements: A survey. *2012 IEEE Students' Conference on Electrical, Electronics and Computer Science*, 1–4.

Barnett, M., Leino, K. R. M., & Schulte, W. (2004). The Spec# programming system: An overview. *International Workshop on Construction and Analysis of Safe, Secure, and Interoperable Smart Devices*, 49–69.

Barthelmess, P., & Anderson, K. M. (2002). A view of software development environments based on activity theory. *Computer Supported Cooperative Work (CSCW)*, *11*(1–2), 13–37.

Bass, L., Clements, P., & Kazman, R. (2013). *Software architecture in practice*. Addison-Wesley Professional.

Bazeley, P. (2009). Analysing qualitative data: More than 'identifying themes.' *Malaysian Journal of Qualitative Research*, *2*(2), 6–22.

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J., & Thomas, D. (2001). *Manifesto for agile software development*. Http://Agilemanifesto.Org/.

Behutiye, W., Karhapää, P., Costal, D., Oivo, M., & Franch, X. (2017). Non-functional requirements documentation in agile software development: Challenges and solution proposal. *International Conference on Product-Focused Software Process Improvement*, 515–522.

Behutiye, W., Karhapää, P., Lopez, L., Burgués, X., Martínez-Fernández, S., Vollmer, A. M., & Oivo, M. (2020). Management of quality requirements in agile and rapid software development: A systematic mapping study. *Information and Software Technology*.

Behutiye, W., Rodríguez, P., Oivo, M., Aaramaa, S., Partanen, J., & Abhervé, A. (2020). How agile software development practitioners perceive the need for documenting quality requirements: A multiple case study. *2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 93–100.

Bevan, N., & Macleod, M. (1994). Usability measurement in context. *Behaviour & Information Technology*, *13*(1–2), 132–145.

Bhatt, P., Ahmad, A. J., & Roomi, M. A. (2016). Social innovation with open source software: User engagement and development challenges in India. *Technovation*,

52, 28–39.

Bhoola, V., Mallik, D., Sharda, A., & Sistu, P. (2013). Measuring software project agility: A special focus on Scrum practices in India. *22nd Australasian Software Engineering Conference: ASWEC 2013*, 27.

Bijan, Y. (2012). *Methodology for engineering requirements for complex systems. (Doctoral dissertation)*. Southern Methodist University, United States of America.

Bloor, M., & Wood, F. (2006). *Keywords in qualitative methods: A vocabulary of research concepts*. Sage.

Boehm, B., & In, H. (1996). Identifying quality-requirement conflicts. *IEEE Software*, *13*(2), 25–35.

Bonuke, O. (2020). *Requirements to reduce software implementation failures in public sector organizations in the United States: A qualitative study. (Doctoral dissertation)*. Colorado Technical University, United States of America.

Bourque, P., & Fairley, R. E. (2014). *Guide to the Software Engineering Body of Knowledge* (3.0). IEEE Computer Society Press.

Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative Research in Psychology*, *3*(2), 77–101.

Braun, V., & Clarke, V. (2018). Thematic analysis. In *Handbook of research methods in health social sciences* (pp. 1–8).

Bridges, D. R. (2018). *Application of multi-criteria decision analysis in requirements definition for prioritization of stakeholder elicitation. (Doctoral dissertation)*. The George Washington University, United States of America.

Bryman, A. (2016). *Social research methods*. Oxford university press.

Camacho, C. R., Marczak, S., & Cruzes, D. S. (2016). Agile team members perceptions on non-functional testing: Influencing factors from an empirical study. *2016 11th International Conference on Availability, Reliability and Security (ARES)*, 582–589.

Cao, L., & Ramesh, B. (2008). Agile requirements engineering practices: An empirical study. *IEEE Software*, *25*(1), 60–67.

Capgemini. (2019). *The World Quality Report 2019-20*. https://www.capgemini.com/wp-content/uploads/2019/10/2019_10_31_WQR_Press-Release.pdf

Carcary, M. (2009). The research audit trial –Enhancing trustworthiness in

qualitative inquiry. *The Electronic Journal of Business Research Methods*, *7*(1), 11 –24.

Castleberry, A., & Nolen, A. (2018). Thematic analysis of qualitative research data: Is it as easy as it sounds? *Currents in Pharmacy Teaching and Learning*, *10*(6), 807–815.

Cervantes, H., & Kazman, R. (2016). *Designing software architectures: A practical approach*. Addison-Wesley Professional.

Chappell, T. W. (2013). *An information systems quandary: Why is there a dearth of interpretive research in a positivist dominated discipline?(Doctoral dissertation)*. Capella University, United States of America.

Chen, L., Babar, M. A., & Nuseibeh, B. (2012). Characterizing architecturally significant requirements. *IEEE Software*, *30*(2), 38–45.

Chillarege, R. (1996). What is software failure? *IEEE Transactions on Reliability*, *45*(3), 354.

Coen, S. M. (2019). *The instructional design of online formative activities: A basic qualitative study*. Capella University, United States of America.

Cope, D. G. (2014). Methods and meanings: credibility and trustworthiness of qualitative research. *In Oncology Nursing Forum*, *41*(1).

Coughlan, J., & Macredie, R. D. (2002). Effective communication in requirements elicitation: a comparison of methodologies. *Requirements Engineering*, *7*(2), 47–60.

Creswell, J. W. (2014). *Research design: Qualitative, quantitative, and mixed methods approaches*. Sage publications.

Creswell, J. W., & Poth, C. N. (2016). *Qualitative inquiry and research design: Choosing among five approaches*. SAGE Publications.

Cumberland, D. M., Sawning, S., Church-Nally, M., Shaw, M. A., Branch, E., & LaFaver, K. (2019). Experiential Learning: Transforming Theory into Practice through the Parkinson's Disease Buddy Program. *Teaching and Learning in Medicine*, *31*(4), 453–465.

Curcio, K., Navarro, T., Malucelli, A., & Reinehr, S. (2018). Requirements engineering: A systematic mapping study in agile software development. *Journal of Systems and Software*, *139*, 32–50.

Dasanayake, S., Aaramaa, S., Markkula, J., & Oivo, M. (2019). Impact of requirements volatility on software architecture: How do software teams keep up with ever- changing requirements? *Journal of Software: Evolution and*

Process.

Davis, A. M. (2004). Achieving quality in software requirements. In *Great Software Debates* (pp. 143–154). Wiley-IEEE Press.

Davis, A. M., & Zowghi, D. (2006). Good requirements practices are neither necessary nor sufficient. *Requirements Engineering*, *11*(1), 1–3.

Deissenboeck, F., Juergens, E., Lochmann, K., & Wagner, S. (2009). Software quality models: Purposes, usage scenarios and requirements. *2009 ICSE Workshop on Software Quality*, 9–14.

Dingsøyr, T., & van Vliet, H. (2009). Introduction to software architecture and knowledge management. In *Software Architecture Knowledge Management* (pp. 1–17). Springer, Berlin, Heidelberg.

Domah, D. (2013). *The NERV methodology: Non-functional requirements elicitation, reasoning and validation in Agile processes. (Doctoral dissertation)*. Nova Southeastern University, United State of America.

Domah, D., & Mitropoulos, F. J. (2015). The NERV methodology: A lightweight process for addressing non-functional requirements in agile software development. *SoutheastCon 2015*, 1–7.

Dragicevic, S., Celar, S., & Novak, L. (2014). Use of method for elicitation, documentation, and validation of software user requirements (MEDoV) in agile software development projects. *2014 Sixth International Conference on Computational Intelligence, Communication Systems and Networks*, 65–70.

Drew, K. (2020). *Government software development project failures: A qualitative approach. (Doctoral dissertation)*. University of Phoenix, United States of America.

Dromey, R. G. (1995). A model for software product quality. *IEEE Transactions on Software Engineering*, *21*(2), 146–162.

Dybå, T., Dingsøyr, T., & Moe, N. B. (2014). Agile project management. In *Software project management in a changing world* (pp. 277–300). Elsevier.

Dybå, T., Prikladnicki, R., Rönkkö, K., Seaman, C., & Sillito, J. (2011). Qualitative research in software engineering. *Empirical Software Engineering*, *16*(4), 425–429.

Eeles, P., Bahsoon, R., Mistrik, I., Roshandel, R., & Stal, M. (2014). Relating system quality and software architecture: Foundations and approaches. In *Relating System Quality and Software Architecture* (pp. 1–20). Morgan Kaufmann.

Elghariani, K., & Kama, N. (2016). Review on Agile requirements engineering

challenges. *2016 3rd International Conference on Computer and Information Sciences (ICCOINS)*, 507–512.

Elijah, J., Mishra, A., Udo, E. M. C., Abdulganiyu, A., & Musa, A. (2017). Survey on requirement elicitation techniques: It's effect on software engineering. *International Journal of Innovative Research in Computer and Communication Engineering*, *5*(5).

Engestrom, Y. (2000). Activity theory as a framework for analyzing and redesigning work. *Ergonomics*, *43*(7), 960–974.

Engeström, Y. (2009). The future of activity theory: A rough draft. In *Learning and expanding with activity theory* (pp. 303–328).

Erumban, A. A., & Das, D. K. (2016). Information and communication technology and economic growth in India. *Telecommunications Policy*, *40*(5), 412–431.

Fair, T. N. (2021). *Strategies to improve project management of software development processes. (Doctoral dissertation)*. Walden University, United States of America.

Farfeleder, S., Moser, T., Krall, A., Stålhane, T., Omoronyia, I., & Zojer, H. (2011). Ontology-driven guidance for requirements elicitation. *Extended Semantic Web Conference*, 212–226.

Farid, W. M. (2011). *The normap methodology: Non-functional requirements modeling for agile processes. (Doctorate dissertation)*. Nova Southeastern University, United States of America.

Farid, W. M. (2012). The Normap methodology: Lightweight engineering of non-functional requirements for agile processes. *2012 19th Asia-Pacific Software Engineering Conference*, 322–325.

Femmer, H., Fernández, D. M., Wagner, S., & Eder, S. (2017). Rapid quality assurance with requirements smells. *Journal of Systems and Software*, *123*, 190–213.

Fernandes, J. C. (2018). *An agile approach for IS/IT benefits management. (Doctoral dissertation)*. University of Lisbon, Portugal.

Fernandes, J. M., & Almeida, M. (2010). Classification and comparison of agile methods. *2010 Seventh International Conference on the Quality of Information and Communications Technology*, 391–396.

Ferrari, A., Spoletini, P., & Gnesi, S. (2016). Ambiguity and tacit knowledge in requirements elicitation interviews. *Requirements Engineering*, *21*(3), 333–355.

Finkelstein, A., & Dowell, J. (1996). A comedy of errors: The London Ambulance

Service case study. *Proceedings of the 8th International Workshop on Software Specification and Design*, 2.

Flick, U. (2018). *An introduction to qualitative research*. Sage Publications Limited.

Flora, H. K., & Chande, S. V. (2014). A systematic study on agile software development methodologies and practices. *International Journal of Computer Science and Information Technologies*, *5*(3), 3626–3637.

Fontes, T. O., & O'Mahony, M. (2008). In-depth interviewing by instant messaging. *Social Research Update*, *53*(2), 1–4.

Fossey, E., Harvey, C., McDermott, F., & Davidson, L. (2002). Understanding and evaluating qualitative research. *Australian and New Zealand Journal of Psychiatry*, *36*(6), 717–732.

Franch, X., Gómez, C., Jedlitschka, A., López, L., Martínez-Fernández, S., Oriol, M., & Partanen, J. (2018). Data-driven elicitation, assessment and documentation of quality requirements in agile software development. *International Conference on Advanced Information Systems Engineering*, 587–602.

Fricker, S. A., Grau, R., & Zwingli, A. (2015). Requirements engineering: Best practice. In *In Requirements Engineering for Digital Health* (pp. 25–46). Springer, Cham.

Fuentes-Fernández, R., Gómez-Sanz, J. J., & Pavón, J. (2010). Understanding the human context in requirements elicitation. *Requirements Engineering*, *15*(3), 267–283.

Gibbs, D. L. (2015). *Constructing requirements: A qualitative study of challenges encountered during requirements elicitation for information systems. (Doctoral dissertation)*. Texas State University.

Gilb, T. (2005). *Competitive engineering: A handbook for systems engineering, requirements engineering, and software engineering using Planguage*. Elsevier.

Gilson, F., Galster, M., & Georis, F. (2019). Extracting quality attributes from user stories for early architecture decision making. *2019 IEEE International Conference on Software Architecture Companion (ICSA-C)*, 129–136.

Glinz, M., & Fricker, S. A. (2015). On shared understanding in software engineering: an essay. *Computer Science-Research and Development*, *30*(3–4), 363–376.

Gorton, I. (2006). *Essential software architecture*. Springer Science & Business Media.

Grady, R. B. (1992). *Practical software metrics for project management and process improvement*. Prentice-Hall, Inc.

Guba, E. G., & Lincoln, Y. (1989). *Fourth generation evaluation*. Sage.

Guttag, J. V., & Horning, J. J. (1993). *Larch: Languages and tools for formal specification*. Springer Science & Business Media.

Heikkila, V. T., Damian, D., Lassenius, C., & Paasivaara, M. (2015). A mapping study on requirements engineering in Agile software development. *2015 41st Euromicro Conference on Software Engineering and Advanced Applications*, 199–207. https://doi.org/10.1109/SEAA.2015.70

Herkert, J., Borenstein, J., & Miller, K. (2020). The Boeing 737 MAX: Lessons for engineering ethics. *Science and Engineering Ethics*, 1–8.

Hess, A., Diebold, P., & Seyff, N. (2019). Understanding information needs of agile teams to improve requirements communication. *Journal of Industrial Information Integration*, *14*, 3–15.

Hickey, A. N. N. M., & Davis, A. M. (2004). A unified model of requirements elicitation. *Journal of Management Information Systems*, *20*(4), 65–84.

Hirshorn, S. R., & Voss, L. D. (2017). *NASA Systems Engineering Handbook*.

Ho, C. W., Johnson, M. J., Williams, L., & Maximilien, E. M. (2006). On Agile performance requirements specification and testing. *AGILE 2006 (AGILE'06)*.

Holvitie, J., Licorish, S. A., Spínola, R. O., Hyrynsalmi, S., MacDonell, S. G., Mendes, T. S., & Leppänen, V. (2018). Technical debt and agile software development practices and processes: An industry practitioner survey. *Information and Software Technology*, *96*, 141–160.

Hull, E., Jackson, K., & Jeremy, D. (2010). *Requirements Engineering*. Springer Science & Business Media.

Huo, M., Verner, J., Zhu, L., & Babar, M. A. (2004). Software quality and Agile methods. *Proceedings of the 28th Annual International Computer Software and Applications Conference, 2004*, 520–525.

Hussain, A., Mkpojiogu, E. O., & Kamal, F. M. (2016). The role of requirements in the success or failure of software projects. *International Review of Management and Marketing*, *6*, 306–311.

Inayat, I., Moraes, L., Daneva, M., & Salim, S. S. (2015). A reflection on agile requirements engineering: solutions brought and challenges posed. *Scientific Workshop Proceedings of the XP2015*, 1–7.

Inayat, I., Salim, S. S., Marczak, S., Daneva, M., & Shamshirband, S. (2015). A systematic literature review on agile requirements engineering practices and challenges. *Computers in Human Behavior*, *51*, 915–929.

ISO/IEC/IEEE. (2011). *29148-2011 - ISO/IEC/IEEE International Standard - Systems and software engineering -- Life cycle processes --Requirements engineering*. https://ieeexplore.ieee.org/document/6146379/

ISO. (2001). *Software Engineering -Product Quality –Part 1: Quality Model*.

Jahja, A. S., Ramalu, S. S., & Razimi, M. S. A. (2021). Generic qualitative research in management studies. *JRAK (Jurnal Riset Akuntansi Dan Bisnis)*, *7*(1), 1–13.

Jain, P., Sharma, A., & Ahuja, L. (2016). ISM based identification of quality attributes for Agile development. *2016 5th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO)*, 615–619.

Jarzabkowski, P., & Wolf, C. (2010). An activity-theory approach to strategy as practice. In *Cambridge handbook of strategy as practice* (pp. 127–140).

Jarzębowicz, A., & Weichbroth, P. (2021). A systematic literature review on implementing non-functional requirements in Agile software development: Issues and facilitating practices. *International Conference on Lean and Agile Software Development*, 91–110.

Jatoba, A., da Cunha, A. M., Vidal, M. C., Burns, C. M., & de Carvalho, P. V. (2019). Contributions from cognitive engineering to requirements specifications for complex sociotechnical systems: A case study in the context of healthcare in Brazil. *Human Factors and Ergonomics in Manufacturing & Service Industries*, *29*(1), 63–77.

Joffe, H. (2012). Thematic analsis. In *Qualitative Research Methods in Mental Health and Psychotherapy: A Guide for Students and Practitioners* (pp. 209–223). John Wiley & Sons.

Jones, C., & Bonsignour, O. (2011). *The economics of software quality*. Addison-Wesley Professional.

Kaplan, B., & Maxwell, J. A. (2005). Qualitative research methods for evaluating computer information systems. In *Evaluating the organizational impact of healthcare information systems*. Springer, New York, NY.

Kasauli, R., Knauss, E., Horkoff, J., Liebel, G., & de Oliveira Neto, F. G. (2021). Requirements engineering challenges and practices in large-scale agile system development. *Journal of Systems and Software*.

Kassab, M. (2014). An empirical study on the requirements engineering practices for

agile software development. *2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications*, 254–261.

Kazman, R., Bianco, P., Ivers, J., & Klein, J. (2020). *Integrability*. https://resources.sei.cmu.edu/asset_files/TechnicalReport/2020_005_001_637385.pdf

Khalid, L. (2019). *Software architecture for business*. Springer.

Khalid, L. (2020). Understanding and dealing with qualities. In *Software Architecture for Business* (pp. 33–50). Springer, Cham.

Kiv, S., Heng, S., Kolp, M., & Wautelet, Y. (2018). Agile manifesto and practices selection for tailoring software development: A systematic literature review. *International Conference on Product-Focused Software Process Improvement*, 12–30.

Kivunja, C., & Kuyini, A. B. (2017). Understanding and applying research paradigms in educational contexts. *International Journal of Higher Education*, *6*(5), 26–41.

Kolb, D. . (1984). *Experiential learning: Experience as the source of learning and development*. Englewood Cliffs, NJ: Prentice Hall.

Kolb, D. A. (2014). *Experiential learning: Experience as the source of learning and development (2nd ed.)*. FT press.

Kolb, D. A., Boyatzis, R. E., & Mainemelis, C. (2001). Experiential learning theory: Previous research and new directions. *Perspectives on Thinking, Learning, and Cognitive Styles*, *1*(8), 227–247.

Kopczyńska, S., Ochodek, M., & Nawrocki, J. (2020). On importance of non-functional requirements in agile software projects—a survey. In *Integrating Research and Practice in Software Engineering* (pp. 145–158). Springer, Cham.

Krasner, H. (2018). *The cost of poor quality software in the US: A 2018 report*.

Krishnamurthy, N., & Saran, A. (2007a). *Building software: A practitioner's guide*. Auerbach Publications.

Krishnamurthy, N., & Saran, A. (2007b). *Building software: A practitioner's guide*. CRC Press.

Kukreja, N. (2015). *Using Social Networking Technology to Improve Collaborative Requirements Elicitation, Negotiation, Prioritization and Evolution*. University of Southern California.

Kumar, M., Shukla, M., & Agarwal, S. (2013). A hybrid approach of requirement

engineering in agile software development. *2013 International Conference on Machine Intelligence and Research Advancement*, 515–519. https://doi.org/10.1109/ICMIRA.2013.108

Kuutti, K. (1996). Activity theory as a potential framework for human-computer interaction research. In *Context and consciousness: Activity theory and human-computer interaction* (p. 1744).

Kyngäs, H. (2020). Qualitative research and content analysis. In *The application of content analysis in nursing science research* (pp. 3–11). Springer, Cham.

Laplante, P. A. (2017). *Requirements engineering for software and systems*. CRC Press.

Lauesen, S. (2002). *Software requirements: Styles and techniques*. Pearson Education.

Leavens, G. T., Baker, A. L., & Ruby, C. (2006). Preliminary design of JML: A behavioral interface specification language for Java. *ACM SIGSOFT Software Engineering Notes*, *31*(3), 1–38.

Leavy, P. (Ed.). (2014). *The Oxford handbook of qualitative research*. Oxford University Press, USA.

Leimane, L., & Nikiforova, O. (2018). Mapping of activities for object-oriented system analysis. *Applied Computer Systems*, *23*(1), 5–11.

Leont'ev, A. N. (1978). *Activity, consciousness, and personality*. Prentice-Hall, Upper Saddle River, NJ.

Lincoln, Y., & Guba, E. (1985). *Naturalistic inquiry*. Beverly Hills, CA: Sage.

Litchmore, K. A. H. (2016). *A comparative study of Agile methods, people factors, and process factors in relation to project success. (Doctoral dissertation)*. Capella University, United States of America.

Liu, L. (2016). Using generic inductive approach in qualitative educational research: A case study analysis. *Journal of Education and Learning*, *5*(2), 129–135.

Lutowski, R. (2016). *Software requirements: Encapsulation, quality, and reuse*. Auerbach Publications.

Lytra, I., Carrillo, C., Capilla, R., & Zdun, U. (2020). Quality attributes use in architecture design decision methods: Research and practice. *Computing*, *102*(2), 551–572.

Maguire, M., & Delahunt, B. (2017). Doing a thematic analysis: A practical, step-by-step guide for learning and teaching scholars. *AISHE-J: The All Ireland Journal*

of Teaching and Learning in Higher Education, *9*(3), 3351–33514.

Maher, C., Hadfield, M., Hutchings, M., & de Eyto, A. (2018). Ensuring rigor in qualitative data analysis: A design research approach to coding combining NVivo with traditional material methods. *International Journal of Qualitative Methods*, *17*(1).

Mairiza, D., Zowghi, D., & Nurmuliani, N. (2009). Managing conflicts among non-functional requirements. *Australian Workshop on Requirements Engineering. University of Technology, Sydney*.

Maiti, R. R., Krasnov, A., Wilborne, D. M., & Mitropoulos, F. (2019). Using OCR to read handwritten texts in search for NFRs in Agile software engineering. *Journal of Software Engineering Practice*, *3*(2), 1–10.

Maiti, R. R., & Mitropoulos, F. J. (2017). Capturing, eliciting, and prioritizing (CEP) NFRs in agile software engineering. *SoutheastCon, 2017*, 1–7.

Malik, M. H., & Velan, N. (2019). Software and services export, IT investment and GDP nexus in India. *International Trade, Politics and Development*, *3*(2), 100–118.

Malik, M. U., Chaudhry, N. M., & Malik, K. S. (2013). Evaluation of efficient requirement engineering techniques in Agile software development. *International Journal of Computer Applications, 83*(3), 24–29.

Malik, M. U., Nasir, H., & Javed, A. (2014). An efficient objective quality model for agile application development. *International Journal of Computer Applications*, *85*(8).

Malone, M. W. (2014). *Process subversion in Agile Scrum software development: A phenomenological approach. (Doctoral dissertation)*. Capella University, United States of America.

Mason, M. (2010). Sample size and saturation in PhD studies using qualitative interviews. *Forum Qualitative Sozialforschung/Forum: Qualitative Social Research*, *11*(3).

Massie, R. A. (2020). *A qualitative study defining behaviors for effective user stories in distributed Scrum teams based in San Antonio, Texas.(Doctoral dissertation)*. Colorado Technical University, United States of America.

Matharu, G. S., Mishra, A., & Singh, H., & Upadhyay, P. (2015). Empirical study of agile software development methodologies: A comparative analysis. *ACM SIGSOFT Software Engineering Notes*, *40*(1), 1–6.

Maxwell, J. A. (2012). *Qualitative research design: An interactive approach*. Sage publications.

McCall, J. A., Richards, P. K., & Walters, G. F. (1977). *Factors in Software Quality, Volumes I, II, and III*.

McCarthy, M. (2016). Experiential learning theory: From theory to practice. *Journal of Business & Economics Research (JBER)*, *14*(3), 91–100.

McLeod, L., MacDonell, S. G., & Doolin, B. (2011). Qualitative research on software development: A longitudinal case study methodology. *Empirical Software Engineering*, *16*(4), 430–459.

Medeiros, J., Vasconcelos, A., Silva, C., & Goulão, M. (2018). Quality of software requirements specification in agile projects: A cross-case analysis of six companies. *Journal of Systems and Software*, *142*, 171–194.

Mendes, T. S., de F. Farias, M. A., Mendonça, M., Soares, H. F., Kalinowski, M., & Spínola, R. O. (2016). Impacts of Agile requirements documentation debt on software projects: A retrospective study. *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, 1290–1295.

Menezes, J., Gusmão, C., & Moura, H. (2019). Risk factors in software development projects: A systematic literature review. *Software Quality Journal*, *27*(3), 1149–1174.

Merriam, S. B. (2009). *Qualitative research: A guide to design and implementation*. John Wiley & Sons.

Merriam, S. B., & Grenier, R. S. (2019). *Qualitative research in practice: Examples for discussion and analysis*. John Wiley & Sons.

Merriam, S. B., & Tisdell, E. J. (2015). *Qualitative research: A guide to design and implementation*. John Wiley & Sons.

Mirsalari, S. R. (2017). *A model for software quality evaluation using the user's point of views. (Doctoral dissertation)*. École Polytechnique de Montréal, Canada.

Moe, N. B., Aurum, A., & Dybå, T. (2012). Challenges of shared decision-making: A multiple case study of agile software development. *Information and Software Technology*, *54*(8), 853–865.

Mohamed, S. F. P. (2015). *A process based approach software certification model for agile and secure environment. (Doctoral dissertation)*. Universiti Utara Malaysia.

Moketar, N. A., & Kamalrudin, M. (2018). Extraction of essential requirements from natural language requirements. *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, *10*(2–2), 35–38.

Morse, W. C., Lowery, D. R., & Steury, T. (2014). Exploring saturation of themes and spatial locations in qualitative public participation geographic information systems research. *Society & Natural Resources*, *72*(5), 557–571.

Nagaria, J., Sadath, L., & Ahmed, S. (2019). Agile implementation-a milestone for academics using software engineering industry practices. *2019 Advances in Science and Engineering Technology International Conferences (ASET)*, 1–6.

Nawrocki, J., Ochodek, M., Jurkiewicz, J., & Kopczyńska, S., Alchimowicz, B. (2014). Agile requirements engineering: A research perspective. *International Conference on Current Trends in Theory and Practice of Informatics*, 40–51.

Nnamchi, J. (2014). *Examining the relationship among systems requirements and information systems project success. (Doctoral dissertation)*. Northcentral University, United States of America.

Nuseibeh, B. (2001). Weaving together requirements and architectures. *Computer*, *34*(3), 115–119.

Nuseibeh, Bashar, & Easterbrook, S. (2000). Requirements engineering: A roadmap. *In Proceedings of the Conference on the Future of Software Engineering*, *1*, 35–46.

O'reilly, M., & Parker, N. (2013). Unsatisfactory Saturation: A critical exploration of the notion of saturated sample sizes in qualitative research. *Qualitative Research*, *13*(2), 190–197.

Ochodek, M., & Kopczyńska, S. (2018). Perceived importance of agile requirements engineering practices–A survey. *Journal of Systems and Software*, 29–43.

Okesola, J., Adebiyi, M., Okokpujie, K., & Adebiyi, A. A. (2019). A systematic literature review of requirement engineering practices in Agile model. *International Journal of Mechanical Engineering and Technology (IJMET)*, *10*(2), 671–687.

Oriol, M., Seppänen, P., Behutiye, W., Farré, C., Kozik, R., Martínez-Fernández, S., & Choras, M. (2019). Data-driven elicitation of quality requirements in agile companies. *International Conference on the Quality of Information and Communications Technology*, 49–63.

Oseni, K., Dingley, K., & Hart, P. (2018). Instant messaging and social networks — The advantages in online research methodology. *International Journal of Information and Education Technology*, *8*(1), 56–62.

Osman, M. H., & Zaharin, M. F. (2018). Ambiguous software requirement specification detection: an automated approach. *In Proceedings of the 5th International Workshop on Requirements Engineering and Testing*, 33–40.

Osorio, K., Rosero, J. L., & Ch, R. P. R. (2020). Technical writer: A proposal to improve quality and documentation in the agile methodology “Scrum”. *KnE Engineering*, 50–59.

Otiji, S. N. (2020). *Strategies managers use to improve software project success and profitability. (Doctoral dissertation)*. Walden University, Unites States of America.

Ozkaya, I., Bass, L., Sangwan, R. S., & Nord, R. L. (2008). Making practical use of quality attribute information. *IEEE Software*, *25*(2), 25–33.

Pacheco, C., García, I., & Reyes, M. (2018). Requirements elicitation techniques: A systematic literature review based on the maturity of the techniques. *IET Software*, *12*(4), 365–378.

Paetsch, F., Eberlein, A., & Maurer, F. (2003). Requirements engineering and Agile software development. *WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003*, 308–313.

Peng, X., Cao, H., Setlur, S., Govindaraju, V., & Natarajan, P. (2013). Multilingual OCR research and applications: An overview. *Proceedings of the 4th International Workshop on Multilingual OCR*, 1–8.

Percy, W. H., Kostere, K., & Kostere, S. (2015). Generic qualitative research in psychology. *The Qualitative Report*, *20*(2), 76–85.

Peterson, J. S. (2019). Presenting a qualitative study: A reviewer’s perspective. *Gifted Child Quarterly*, *63*(3), 147–158.

Phalnikar, R., Deshpande, V. S., & Joshi, S. D. (2009). Applying Agile principles for distributed software development. *International Conference on Advanced Computer Control*, 535–539.

Pham, A. (2011). *Scrum in Action: Agile software project management and development*. Cengage Learning.

Pierce, S. P. (2018). *Web 2.0 in the online learning environment: A basic qualitative study to define best practices. (Doctoral dissertation)*. Northcentral University, United States of America.

Pietkiewicz, I., & Smith, J. A. (2014). A practical guide to using interpretative phenomenological analysis in qualitative research psychology. *Psychological Journal*, *20*(1), 7–14.

Pinto, T. D., Gonçalves, W. I., & Costa, P. V. (2019). User interface prototype generation from agile requirements specifications written in concordia. *Proceedings of the 25th Brazillian Symposium on Multimedia and the Web*, 61–

64.

Pivec, M., Trummer, C., & Pripfl, J. (2007). Usable collaborative email requirements using activity theory. *Informatica*, *3*(1), 71–83.

Poort, E. R., Martens, N., Van De Weerd, I., & Van Vliet, H. (2012). How architects see non-functional requirements: Beware of modifiability. *International Working Conference on Requirements Engineering: Foundation for Software Quality*, 37–51.

Poppendieck, M., & Poppendieck, T. (2003). *Lean software development: An Agile toolkit*. Addison-Wesley.

Pour-Previti, M. (2019). *Engaging experience: Experiential learning & student engagement in online community college courses. (Doctoral dissertation)*. United Stated of America, University of West Georgia.

Pressman, R. S. (2010). *Software engineering: A practitioner's approach* (7th ed.). Palgrave Macmillan.

Quinn Patton, M. (1990). *Qualitative research and evaluation methods*. Sage.

Radulovic, F., & García-Castro, R. (2011). Towards a quality model for semantic technologies. *International Conference on Computational Science and Its Applications*, 244–256.

Ramesh, B., Cao, L., & Baskerville, R. (2010). Agile requirements engineering practices and challenges: an empirical study. *Information Systems Journal*, *20*(5), 449–480.

Ramos, F. B. A., Costa, A. A. M., Perkusich, M., Almeida, H. O., & Perkusich, A. (2018). A non-functional requirements recommendation system for scrum-based projects. *SEKE*, 149–148.

Read, A. (2013). *Improving requirements generation thoroughness in user-centered workshops: The role of prompting and shared user stories. (Doctoral dissertation)*. University of Nebraska, United States of America.

Riaz, M. Q., Butt, W. H., & Rehman, S. (2019). Automatic detection of ambiguous software requirements: An insight. *2019 5th International Conference on Information Management (ICIM)*, 1–6.

Rivero, J. M., Grigera, J., Distante, D., Montero, F., & Rossi, G. (2019). DataMock: An Agile approach for building data models from user interface mockups. *Software & Systems Modeling*, *18*(1), 663–690.

Robertson, S., & Robertson, J. (2012). *Mastering the requirements process: Getting requirements right*. Addison-Wesley.

Robiolo, G., Scott, E., Matalonga, S., & Felderer, M. (2019). Technical debt and waste in non-functional requirements documentation: An exploratory study. *International Conference on Product-Focused Software Process Improvement*, 220–235.

Rodrigues, P., Ecar, M., Menezes, S. V., da Silva, J. P. S., Guedes, G. T., & Rodrigues, E. M. (2018). Empirical evaluation of formal method for requirements specification in Agile approaches. *Proceedings of the XIV Brazilian Symposium on Information Systems*, 1–8.

Ronen, B. (2017). *Excessive software development : Practices and penalties*. *35*, 13–27.

Rossberg, J. (2019). Introduction to Scrum and Agile Concepts. In *Agile Project Management with Azure DevOps* (pp. 67–123). Apress, Berkeley, CA.

Rossman, G. B., & Rallis, S. F. (2016). *An introduction to qualitative research: Learning in the field*. Sage Publications.

Rubin, K. S. (2012). *Essential Scrum: A practical guide to the most popular Agile process*. Addison-Wesley.

Sabry, A. E., & El-Rabbat, S. S. (2015). Proposed framework for handling architectural nfr's within scrum methodology. *Proceedings of the International Conference on SERP*, 238.

Sagheer, M., Zafar, T., & Sirshar, M. (2015). A framework for software quality assurance using agile methodology. *International Journal of Scientific & Technology Research*, *4*(2), 44–50.

Sajid, A., Nayyar, A., & Mohsin, A. (2010). Modern trends towards requirement elicitation. *In Proceedings of the 2010 National Software Engineering Conference*, 9.

Saldaña, J. (2015). *The coding manual for qualitative researchers*. Sage.

Saleh, M., Baharom, F., Mohamed, S. F. P., & Din, A. M. (2021). The current practices and required knowledge for non-functional requirements elicitation in Agile context: A pilot study in Jordan. *Proceedings of Knowledge Management International Conference (KMICe) 2021*, 265–270.

Sarma, S. K. (2015). Qualitative research: Examining the misconceptions. *South Asian Journal of Management*, *22*(3), 176–191.

Saunders, B., Sim, J., Kingstone, T., Baker, S., Waterfield, J., Bartlam, B., & Jinks, C. (2018). Saturation in qualitative research: exploring its conceptualization and operationalization. *Quality & Quantity*, *52*(4), 1893–1907.

Schön, E. M., Sedeño, J., Mejías, M., Thomaschewski, J., & Escalona, M. J. (2019). A metamodel for Agile requirements engineering. *Journal of Computer and Communications*, *7*, 1–22.

Seaman, C. B. (1999). Qualitative methods in empirical studies of software engineering. *IEEE Transactions on Software Engineering*, *25*(4), 557–572.

Seaman, Carolyn B. (1999). Qualitative methods in empirical studies of software engineering. *IEEE Transactions on Software Engineering*, *25*(4), 557–572.

Shahid, M., & Tasneem, K. A. (2017). Impact of avoiding non-functional requirements in software development stage. *Am. J. Inf. Sci. Comput. Eng*, *3*(4), 52–55.

Shanmugapriya, P., & Kumaran, N. (2016). Predominant quality attributes in evaluating software architecture and addressing scenario coverage problem. *International Journal of Advanced Research in IT and Engineering*, *5*(5), 1–7.

Shenton, A. K. (2004). Strategies for ensuring trustworthiness in qualitative research projects. *Education for Information*, *22*(2), 63–75.

Singh, A. (2012). Agile: Analysis of its problems and their solutions. *IJCA Proc Int Conf Recent Adv Future Trends Inf Technol*, 32–35.

Sinpang, J. S., Sulaiman, S., & Idris, N. (2017). Detecting ambiguity in requirements analysis using Mamdani fuzzy inference. *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, *9*(3–4), 157–162.

Smith, G. (2012). *The Object-Z specification language*. Springer Science & Business Media.

Soares, H. F., Alves, N. S., Mendes, T. S., Mendonça, M., & Spínola, R. O. (2015). Investigating the link between user stories and documentation debt on software projects. *2015 12th International Conference on Information Technology-New Generations*, 385–390.

Solinski, A., & Petersen, K. (2016). Prioritizing Agile benefits and limitations in relation to practice usage. *Software Quality Journal*, *24*(2), 447–482.

Sommerville, I. (2015). *Software engineering*. Pearson Education Limited.

Spivey, J. (1992). *The Z notation: A reference manual*. Prentice Hall International.

Ssbriye, A. O. J. A., & Zainon, W. M. N. W. (2018). An approach for detecting syntax and syntactic ambiguity in software requirements specification. *Journal of Theoretical & Applied Information Technology*, *96*(8).

Stake, R. E. (2010). *Qualitative research: Studying how things work*. Guilford Press.

Stieger, S., & Göritz, A. S. (2006). Using instant messaging for Internet-based interviews. *CyberPsychology & Behavior*, *9*(5), 552–559.

Suma, V., & Shubhamangala, B. R. (2013). A comprehensive analysis of factors influencing quality of requirements. *Lecture Notes on Software Engineering*, *1*(2), 199.

Svensson, A. (2020). Identifying motives for implementing eHealth by using activity theory. *Sustainability*, *12*(4), 1298.

Tabassum, A., Bhatti, D. S. N., Asghar, A. R., Manzoor, I., & Alam, I. (2017). Optimized quality model for Agile development: Extreme programming (XP) as a case scenario. *International Journal of Advanced Computer Science and Applications*, *8*(4), 392–400.

Tenório, N., Pinto, D., Silva, M. J., de Almeida, I. C., & Bortolozzi, F. (2020). Knowledge management in the software industry: How Scrum activities support a knowledge management cycle. *Navus-Revista de Gestão e Tecnologia, 10*, 1–13.

Tracy, S. J. (2019). *Qualitative research methods: Collecting evidence, crafting analysis, communicating impact*. John Wiley & Sons.

Tripathi, V., & Goyal, A. K. (2014). Agile requirement engineer: Roles and responsibilities. *International Journal of Innovative Science, Engineering & Technology*, *1*(3), 213–219.

Tripp, J. F. (2012). *The impacts of Agile development methodology use on project success: A contingency view (Doctoral dissertation)*. Michigan State University, United States of America.

Tukur, M., Umar, S., & Hassine, J. (2021). Requirement engineering challenges: A systematic mapping study on the academic and the industrial perspective. *Arabian Journal for Science and Engineering*, 1–26.

Uteliyeva, N. K., Ismail, E. E., & Akhmetov, D. F. (2019). Comparative analysis of models of quality of software tools. *Journal of Mathematics, Mechanics and Computer Science*, *104*(4), 85–94.

Vermeulen, D., Weger, M. D., & Ravesteyn, P. (2017). Focus on non-functionals! The effect of focussing on non-functional requirements on the maturity of the requirements engineering process. *Communications of the IIMA*, *15*(1), 5.

VersionOne. (2018). *13th annual state of Agile survey report*. https://stateofagile.com/#ufh-i-613553418-13th-annual-state-of-agile-report/7027494

Vierhauser, M., Cleland-Huang, J., Burge, J., & Grünbacher, P. (2019). The

interplay of design and runtime traceability for non-functional requirements. *Proceedings of the 10th International Workshop on Software and Systems Traceability*, 3–10.

Vygotsky, L. (1978). *Mind in society: The development of higher psychological processes*. Harvard University Press, Cambridge, Mass.

Wagner, S., Fernández, D. M., Felderer, M., & Kalinowski, M. (2017). Requirements engineering practice and problems in agile projects: Results from an international survey. *Proc.of the 20th Ibero-American Conference on Software Engineering (CIBSE), Requirements Engineering (WER) Track, Buenos Aires, Argentina, 2017*.

Wagner, S., Méndez-Fernández, D. Kalinowski, M., & Felderer, M. (2018). Agile requirements engineering in practice: Status quo and critical problems. *CLEI Electronic Journal*, *21*(1).

Walden, D. D., Roedler, G. J., Forsberg, K., Hamelin, R. D., & Shortell, T. M. (2015). *Systems engineering handbook: A guide for system life cycle processes and activities*. John Wiley & Sons.

Wang, X., Zhao, L., Wang, Y., & Sun, J. (2014). The role of requirements engineering practices in agile development: An empirical study. In *Requirements engineering* (pp. 195–209). Springer, Berlin, Heidelberg.

Werner, C., Li, Z. S., Ernst, N., & Damian, D. (2020). The lack of shared understanding of non-functional requirements in continuous software engineering: Accidental or essential? *2020 IEEE 28th International Requirements Engineering Conference (RE)*, 90–101.

Wiegers, K., & Beatty, J. (2013). *Software requirements*. Pearson Education.

Yin, R. K. (2011). *Qualitative research from start to finish*. Guilford Publications.

You, J., Li, J., & Xia, S. (2012). A survey on formal methods using in software development. *IET International Conference on Information Science and Control Engineering 2012 (ICISCE 2012), Shenzhen*, 1–4.

Younas, M., Jawawi, D. N. A., Shah, M. A., Mustafa, A., Awais, M., Ishfaq, M. K., & Wakil, K. (2020). Elicitation of nonfunctional requirements in agile development using cloud computing environment. *IEEE Access*, 209153–209162.

Zamudio, L., Aguilar, J. A., Tripp, C., & Misra, S. (2017). Requirements engineering techniques review in Agile software development methods. *In International Conference on Computational Science and Its Applications*, 683–698.

Zhu, H. (2005). *Software design methodology: From principles to architectural*

styles. Elsevier.

Zowghi, D., & Coulin, C. (2005). Requirements elicitation: A survey of techniques, approaches, and tools. In *Engineering and managing software requirements* (pp. 19–46). Springer, Berlin, Heidelberg.

Appendix A

First Version of Interview Protocol

- **A heading**

1. Participant Name:
2. Participant Gender:
3. Participant Number:
4. Date of Interview:
5. Interview Duration:

- **Background Questions**

1. Please, what is your role based on Scrum Agile method (e.g., product owner, Scrum master or software developer)?
2. Please, how many years of experience in eliciting quality attributes from customers and users based on Scrum Agile method?
3. Please, what are the projects types (e.g., onshore and offshore) that you involved in during your experience in eliciting quality attributes from customers and users based on Scrum Agile method?
4. Please, what is the size of your software company (e.g., big, medium, small)?

- **Interview Questions**
  1. Please, can you tell me how do you elicit (i.e., collect, gather, define) quality attributes from customers and users?
  2. Please, to what extent customers and users are familiar with quality attributes concepts and have the capability to discuss them?
  3. Please, how do you manage to overcome neglect quality attributes and ensure paying attention to all quality attributes during requirements elicitation?
  4. Please, how do you manage to achieve common understanding of quality attributes with customers and users?
  5. Please, how do you document quality attributes during requirements elicitation?

- **Interview Instructions**
  1. Start the interview by greeting the participant, ensure receiving the Informed Consent Form and download the Informed Consent Form to your computer.
  2. Establish warm-up discussion with a participant and build rapport with the participant. Moreover, encourage the participant to state positive and negative experiences as well that can help to know any challenges or benefits related to eliciting quality attributes in Scrum.

3. After the warm-up discussion, start asking the participants about their background information from interview protocol.
4. After finishing the background questions, start asking the participants the interview questions from the interview protocol. Moreover, use initial probes from probes table to gain additional information.
5. Ask the participant any additional probes when it becomes necessary to gain in-depth explanation about the research phenomena.
6. Before finishing the interview, review the questions and answers to ensure covering of all necessary questions.
7. Request from the participant to review the findings of the interview analysis after finishing the analysis.
8. When you finish the interview, express your gratitude to the participant and appreciate his efforts and cooperation for involving in the interview.
9. Transfer the answers of background questions from instant messaging box to the document of participant profile.
10. Transfer the answers of all interview questions from instant messaging box to the document of interview transcript.

Appendix B
Second Version of Interview Protocol

- **A heading**

1. Participant Name:
2. Participant Gender:
3. Participant Number:
4. Date of Interview:
5. Interview Duration:

- **Background Questions**

1. Please, what is your role based on Scrum Agile method (e.g., product owner, Scrum master or software developer)?
2. Please, how many years of experience in eliciting quality attributes from customers and users based on Scrum Agile method?
3. Please, what are the projects types (e.g., onshore and offshore) that you involved in during your experience in eliciting quality attributes from customers and users based on Scrum Agile method?
4. Please, what is the size of your software company (e.g., big, medium, small)?

- **Interview Questions**

1. Please, can you tell me how do you elicit (i.e., collect, gather, define) quality attributes from customers and users?
2. Please, to what extent customers and users are familiar with quality attributes concepts and have the capability to discuss them?
3. Please, how do you manage to overcome neglect quality attributes and ensure paying attention to all quality attributes during requirements elicitation?
4. Please, how do you manage to achieve common understanding of quality attributes with customers and users?
5. Please, how do you document quality attributes during requirements elicitation?

- **Probes from Pilot Interview**

1. Please, specify who attend the discussion of quality attributes?
2. Please, can you explain the roles of Scrum team members during the interaction with customers and users?
3. Please, to what extent you invite users to participate in elicitation of quality attributes?

- **Interview Instructions**

1. Start the interview by greeting the participant, ensure receiving the Informed Consent Form and download the Informed Consent Form to your computer.

2. Establish warm-up discussion with a participant and build rapport with the participant. Moreover, encourage the participant to state positive and negative experiences as well that can help to know any challenges or benefits related to eliciting quality attributes in Scrum.
3. During this rapport, encourage the participant to associate his answers and perspectives with real examples from his projects as much as he can when he answers any question.
4. After the warm-up discussion, start asking the participants about their background information from interview protocol.
5. After finishing the background questions, start asking the participants the interview questions from the interview protocol. Moreover, use initial probes from probes table to gain additional information.
6. Ask the participant any additional probes when it becomes necessary to gain in-depth explanation about the research phenomena.
7. Before finishing the interview, review the questions and answers to ensure covering of all necessary questions.
8. Request from the participant to review the findings of the interview analysis after finishing the analysis.
9. When you finish the interview, express your gratitude to the participant and appreciate his efforts and cooperation for involving in the interview.
10. Transfer the answers of background questions from instant messaging box to the document of participant profile.
11. Transfer the answers of all interview questions from instant messaging box to the document of interview transcript.

Appendix C

Third Version of Interview Protocol

- **A heading**

1. Participant Name:
2. Participant Gender:
3. Participant Number:
4. Date of Interview:
5. Interview Duration:

- **Background Questions**

1. Please, what is your role based on Scrum Agile method (e.g., product owner, Scrum master or software developer)?
2. Please, how many years of experience in eliciting quality attributes from customers and users based on Scrum Agile method?
3. Please, what are the projects types (e.g., onshore and offshore) that you involved in during your experience in eliciting quality attributes from customers and users based on Scrum Agile method?
4. Please, what is the size of your software company (e.g., big, medium, small)?

- **Interview Questions**

1. Please, can you tell me how do you elicit (i.e., collect, gather, define) quality attributes from customers and users?
2. Please, to what extent customers and users are familiar with quality attributes concepts and have the capability to discuss them?
3. Please, how do you manage to overcome neglect quality attributes and ensure paying attention to all quality attributes during requirements elicitation?
4. Please, how do you manage to achieve common understanding of quality attributes with customers and users?
5. Please, how do you document quality attributes during requirements elicitation?

- **Probes from Pilot Interview**

1. Please, specify who attend the discussion of quality attributes?
2. Please, can you explain the roles of Scrum team members during the interaction with customers and users?
3. Please, to what extent you invite users to participate in elicitation of quality attributes?

- **Probes during Interview with Participants**

1. Please, can you give more details to describe what happens through the interaction with customers and users during elicitation of quality attributes?
2. Please, can you give additional details about the structure of questionnaire that you follow to gain information about quality attributes?
3. Please, in the case of offshore projects, how do you communicate with customers and users?
4. Please, can you explain what do you do to be knowledgeable of quality attributes domain and what advantages you can get from knowing quality attributes domain?
5. Please, what is the importance and impact of documentation of quality attributes?

6. Please, based on your answers, can you describe the chronology of your actions and guidelines (i.e. sequence and arrangement of what you do) regarding eliciting unambiguous quality attributes?

- **Interview Instructions**

1. Start the interview by greeting the participant, ensure receiving the Informed Consent Form and download the Informed Consent Form to your computer.
2. Establish warm-up discussion with a participant and build rapport with the participant. Moreover, encourage the participant to state positive and negative experiences as well that can help to know any challenges or benefits related to eliciting quality attributes in Scrum.
3. During this rapport, encourage the participant to associate his answers and perspectives with real examples from his projects as much as he can when he answers any question.
4. After the warm-up discussion, start asking the participants about their background information from interview protocol.
5. After finishing the background questions, start asking the participants the interview questions from the interview protocol. Moreover, use initial probes from probes table to gain additional information.
6. Ask the participant any additional probes when it becomes necessary to gain in-depth explanation about the research phenomena.
7. Before finishing the interview, review the questions and answers to ensure covering of all necessary questions.
8. Request from the participant to review the findings of the interview analysis after finishing the analysis.
9. When you finish the interview, express your gratitude to the participant and appreciate his efforts and cooperation for involving in the interview.
10. Transfer the answers of background questions from instant messaging box to the document of participant profile.
11. Transfer the answers of all interview questions from instant messaging box to the document of interview transcript.

Appendix D
Informed Consent Form

COLLEGE OF ARTS AND SCIENCES

UNIVERSITI UTARA MALAYSIA

INFORMED CONSENT FORM

Exploring the Practices and the Process of Eliciting Unambiguous Quality Attributes in Scrum-based Projects

Dear Respected Participant,

This is a consent form for the participants who agree to participate in an interview to explore the practices and the process of eliciting unambiguous quality attributes in Scrum-based projects. You are invited to participate in this study because you fulfil the following four criteria: knowledge of quality attributes, experience in eliciting unambiguous quality attributes from customers and users for more than 3 years, following Scrum Agile method in your projects and spent your experience in Indian software organization. If you decide to participate, the researcher will ask you to sign this consent form. The purpose of this study is to explore the practices and the process that can help to elicit unambiguous quality attributes from customers and users in projects that follow Scrum Agile method. Your participation will provide

significant help to develop high-quality software in the market and enlighten academic research regarding this important topic. The expected duration of the interview will be approximately 2 hours. The interview will be conducted based on your free time and via instant messaging. Your confidentiality is of high concern and it will be highly protected and maintained. The results of this study may be published, but your name will not be presented to protect your privacy. Your participation in this study is voluntary. There is no compulsion to complete the interview; you have the right to withdraw from the interview at any time.

Please, if you have any questions regarding this study, please send a message to the researcher: Hussin Ahmed at hussin_ahmed@ahsgs.uum.edu.my or the supervisor: Prof. Madya Dr. Azham Bin Hussain at azham.h@uum.edu.my. Thank you.

I have read the consent form. I understand the information and I agree to participate in this study.

Name of Participant: .....

Signature:..... Date: .....

Appendix E
Samples from Analytic Memo

1- Section 1: Interview Questions and Answers

- **Sample from Participant 1**

Question: Please, can you tell me how do you elicit (i.e., collect, gather, define) quality attributes from customers and users?

Answer: *I would like to say that eliciting quality attributes is part of our responsibility for developing high-quality software. In the very beginning, I conduct a meeting with customer to ask him about important matters for future discussion of requirements. The most important matters in the beginning are to know quality attributes domain and crucial challenges that need to be solved. Then, I conduct a meeting for discussing the required requirements where project team, customer and some users attend the discussion to ask questions such as the common problems that users face during executing any function or the required response time for certain functionality. Also, I can show previous user interfaces from previous projects to customer and users during the discussion of quality attributes which were prepared before the discussion. In most cases, I request from software designer to prepare a portfolio of relevant user interfaces design to use them in elicitation of requirements.*

- **Sample from Participant 2**

Question: Please, can you explain what do you do to be knowledgeable of quality attributes domain and what advantages you can get from knowing quality attributes domain?

Answer: *In our company we check software market which can be a guide to know the quality attributes domain. The advantages of knowing domain include covering all necessary requirements which may be forgotten during the elicitation of quality attributes and familiarization with common concepts in quality attributes domain to prevent any misunderstanding of these concepts from the side of development team.*

- **Sample from Participant 3**

Question: Please, can you give additional details about the structure of questionnaire that you follow to gain information about quality attributes?

Answer: *The survey or questionnaire contains questions about important services that will be provided by the system, important qualities, and deadline for finishing the project. As I mentioned before, we only use this questionnaire to provide us top level information about quality attributes before second session with customer and Scrum team members.*

- **Sample from Participant 4**

Question*:* Please, can you give more details to describe what happens through the interaction with customers and users during elicitation of quality attributes?

Answer: *I share examples with customer of other portals that faced challenges like Flipkart, a popular Indian website that suffered from downtime. Sharing these cases of low-quality software is important for two reasons. The first reason is considering the cost to test quality attributes because we may hire testers from outside the company and the second reason is considering sufficient time for development and testing quality attributes. The key guidance in interviewing customer is using understood terms. So, I need to define these terms in a way that can be understood by non-technical people. Using terms that are not popular or normal to customer can act as a barrier to understanding quality attributes. For instance, while Scrum team considers using vulnerability a normal word in our technical conversation when we talk about security, it may not be normal to the customer.*

- **Sample from Participant 5**

Question: Please, to what extent customers and users are familiar with quality attributes concepts and have the capability to discuss them?

Answer: *"As I said before, customers and users are not technical people and they do not have technical background like software developers. So, our knowledge of their domain is important to get over this issue before we make them tell us about their quality attributes .It is also important to inform them about the importance of quality attributes. This can help to allocate additional budget for making sure that quality attributes are complete".*

- **Sample from Participant 6**

Question: Please, how do you manage to achieve common understanding of quality attributes with customers and users?

Answer: *"Since there are hundreds of systems in the market, I can capitalize on this advantage to let customer select from these systems functionalities and qualities that are quite similar to his expectation. These systems make requirements of customers obvious, clear, understandable and less ambiguous. Additionally, these systems in the market cover plenty of requirements which encourage customers to discuss what they not discussed before during the interview".*

- **Sample from Participant 7**

Question: Please, how do you document quality attributes during requirements elicitation?

Answer: "*I focus on defining what I have to do as a software developer for making a function usable and reliable without failure. I also focus on defining function name, users of the function and actions of users like loading a file or downloading a file. If downloading file is finished completely, I let the user know that downloading a file was complete. Sometimes, it is possible to lose connection with database, so, I let the user know that he needs to try downloading the file again".*

- **Sample from Participant 8**

Question: Please, what is the importance and impact of documentation of quality attributes.

Answer: "*When I start development with other developers, documentation of requirements guides us to consider what we may forget. Moreover, documenting requirements assists Scrum team to show these requirements to customers in future projects which are similar to the one we developed before".*

2- Section 2: Familiarization with Data

- **Sample from Participant 1**

It is interesting that Participant 1 has shown her positive attitude towards elicitation of quality attributes. She takes elicitation of quality attributes seriously as she referred in her first answer that "*I would like to say that eliciting quality attributes is part of our responsibility for developing high-quality software*". This is important in software industry to make software projects successful.

She posited that she conducts a meeting with customer for the purpose of receiving answers about significant matters from her perspective as she said "*In the very beginning, I conduct a meeting with customer to ask him about important matters for future discussion of requirements*". According to her view, these matters consist of knowing exactly the domain of interest and the problems that challenged customer in his company as she said "*The most important matters in the beginning are to know quality attributes domain and crucial challenges that need to be solved*". Of course, identifying domain is important to know the scope of project and identifying challenges is important to solve them.

She also demonstrated that she conducted additional meeting to ask several questions to customers and users to know their requirements as she said **"***Then, I conduct a meeting for discussing the required requirements where project team, customer and some users attend the discussion to ask questions such as the common problems that users face during executing any function or the required response time for certain functionality***".**

She also referred to using user interfaces during the discussion with customer and users as she noted **"***Also, I can show previous user interfaces from previous projects to customer and users during the discussion of quality attributes which were prepared before the discussion***".** Furthermore, she referred to her request from software designer in her Scrum team to prepare a group or collection of user interfaces as she said **"***I request from software designer to prepare a portfolio of relevant user interfaces design to use them in elicitation of requirements***".** This is interesting matter to take elicitation seriously and prepare what can help in elicitation of requirements.

- **Sample from Participant 2**

Participant 2 has defined the way to be knowledgeable about quality attributes which is checking software market. This checking possibly means to know the features of domain by launching some software from the market and understanding their features. As he said "*In our company we check software market which can be a guide to know the quality attributes domain*".

He stated the advantages he can get from being knowledgeable about domain which contains two advantages. The first advantage is avoid forgetting any requirement as he said "*The*

advantages of knowing domain include covering all necessary requirements which may be forgotten during the elicitation of quality attributes" and the second advantage is achieving common understanding of domain concepts as he said "*familiarization with common concepts in quality attributes domain to prevent any misunderstanding of these concepts from the side of development team*". This means that this familiarization of domain concepts helps to reduce ambiguity.

- **Sample from Participant 3**

From the answer of Participant 3, the structure of questionnaire consists of three sections or three type of questions related to the services, qualities and deadline of the project. As he said "*The survey or questionnaire contains questions about important services that will be provided by the system, important qualities, and deadline for finishing the project.*" The services possibly represent the functions of software that will provide to users and qualities represent quality attributes like usability and reliability. Deadline of finishing the project is possibly important to know when exactly they will deliver the final version of software.

The participant referred again to the importance of conducting additional session with customer in addition to sending questions in a questionnaire as he said "*As I mentioned before, we only use this questionnaire to provide us top level information about quality attributes before second session with customer and Scrum team members"*. This means that questionnaire precedes another session with customers and users.

- **Sample from Participant 4**

Participant 4 added additional details about what happens during the interaction with customers to know quality attributes. It is very interesting that she referred to sharing examples of software portals or website which had low-quality. She mentioned specific website which is Flipkart as she said "*I share examples with customer of other portals that faced challenges like Flipkart, a popular Indian website that suffered from downtime*".

According to her explanation, she aims to make customer understand that there is cost associated for developing high-quality software. Furthermore, she wants to make customer understand that enough time is needed. This means that she does not want customer to put them under pressure to finish quickly the development of software as she said "*The first reason is considering the cost to test quality attributes because we may hire testers from outside the company and the second reason is considering sufficient time for development and testing quality attributes*".

She also referred to an important guideline which is defining terms for customers to make communication effective as she said "*The key guidance in interviewing customer is using understood terms. So, I need to define these terms in a way that can be understood by non-technical people. Using terms that are not popular or normal to customer can act as a barrier to understanding quality attributes"*. She gave example of these terms which is vulnerability that maybe not understood by non-technical people as she said *"For instance, while Scrum team considers using vulnerability a normal word in our technical conversation when we talk about security, it may not be normal to the customer"*. I think this guidance has significant value to reduce ambiguity and need to be taken into consideration.

- **Sample from Participant 5**

Participant 5 mentioned again that based on her experience that customers and users do not have the technical knowledge like any professional in software industry when she said "*As I said before, customers and users are not technical people and they do not have technical background like software developers*".

From her perspective, the ideal way to overcome this limited technical knowledge is being knowledgeable about possible quality attributes in the domain. She said "*So, our knowledge of their domain is important to get over this issue before we make them tell us about their quality attributes*". This quote refers to the necessity to prepare seriously before meeting customers or users to guide them. This means that Scrum team have to be active and serious in preparation for ultimate guidance to customers and users to elicit quality attributes from them.

Furthermore, the Participant 5 referred to the significance of teaching customer that quality attributes are important as she said "*It is also important to inform them about the importance of quality attributes. This can help to allocate additional budget for making sure that quality attributes are complete*". The reason behind this act is to encourage customer to allocate enough money for meeting and fulfilling these quality attributes.

- **Sample from Participant 6**

Participant 6 has mentioned and focused on a guideline that helps Scrum team to achieve common understanding and reduce ambiguous requirements. This guidance is using systems in the market as he said **"***Since there are hundreds of systems in the market, I can capitalize on this advantage to let customer select from these systems functionalities and qualities that are quite similar to his expectation*".

He emphasized the value that can result from this guidance in terms to making requirements clear to Scrum team and sure that developing these requirements will meet the expectations of customers and users as he said "*These systems make requirements of customers obvious, clear, understandable and less ambiguous.*" The quote also refers to another value which is discussing more requirements which maybe forgotten before to mention by customer during the discussion as he said "*Additionally, these systems in the market cover plenty of requirements which encourage customers to discuss what they not discussed before during the interview*".

- **Sample from Participant 7**

Participant 7 has shown his approach for documenting quality attributes and focused on the importance of documenting the required response to achieve quality attributes such as making a function usable or reliable as he said "*I focus on defining what I have to do as a software developer for making a function usable and reliable without failure*".

Moreover, he represents key elements of function like function name, actions, and actors as he said ". *I also focus on defining function name, users of the function and actions of users like loading a file or downloading a file*".

- **Sample from Participant 8**

Participant 8 acknowledged the benefit from documentation is twofold. The first benefit is remembering information that can be guidance for them during software development as he said "*When I start development with other developers, documentation of requirements guides us to consider what we may forget".* This is natural because developers need written description of what will be developed, otherwise, they will assume requirements that maybe not the actual requirements that needed by customers and users.

Furthermore, he said that documentation has another benefit which is using documentation again in future projects *"Moreover, documenting requirements assists Scrum team to show these requirements to customers in future projects which are similar to the one we developed before*". Of course, the projects have to be similar or maybe have some common functions like login functionality that have similar quality attributes like security and usability.

- **Sample of Memoing during Familiarization of data**

- I have to clearly discover or notice what actions or guidelines participants do or follow? When they do these actions or follow these guidelines? Why or for what reasons?
- I think that some participants set an introductory appointment of quality attributes. From the answers of five participants, participant1, participant 2, participant 5, participant 7 and participant 8 they do similar action or follow a same guideline which is having introductory or initial or first session of interview with customer. This action of these participants can raise a question: why do they need this first session?
- I think this earlier discussion of these of five participants means that the participants are serious about elicitation of unambiguous quality attributes and they need to be organized and involve in the discussion with prior knowledge to effectively elicit unambiguous quality attributes. This prior knowledge leads them to know the exact domain, the business goal, end users and software type.
- I can see from some quotes reasons for having introductory session. An interesting reason is checking their previous projects to see what they have done or developed in the past in other software that maybe similar to the one they will develop. Another interesting reason is downloading similar projects to the one they will develop with customer. This attitude means that these participants are professional and they are interested to cover what needs to elicit unambiguous quality attributes.
- I discovered that other participants seem to set interview immediately without preparation or conducting interview in advance. From the answers of three participants: participant 3, participant 4 and participant 6 they set interview directly with customers and users. But there is a difference between these participants that I have to mention, remember and emphasize. This difference is sending questionnaire or a collection of questions to customer by two participants: participant 3 and participant 6. This means that two participants: participant 3 and participant 6 also do an action before setting another interview.
- From my reading, I think that the purpose of sending these questions in questionnaire is having a prior understanding or prior coverage of requirements that include functional

and quality attributes. I must understand the structure of these questionnaires and review the answers of these two participants who use questionnaire.

- I found significant matter that makes me astonished which is appreciation of participant 1, participant 2 and participant 8 to software reuse. I assume that they don't want to waste their time in designing or developing what they already have designed before. But, they consider that reusing these designs will not be blindly but with consulting customers and users. I mean they discuss with customers and users whether customer and users accept these designs or not.
- I have to remember and highlight the attitude of all participants towards their seriousness of eliciting unambiguous quality attributes. I can see clearly they take elicitation seriously and try to do what it takes to ensure developing quality software.
- I noticed significant guidelines to elicit unambiguous quality attributes like mock-ups, software in the market that are similar to the software being developed, prototypes and previous user interface design. I have to pay attention to these guidelines because they are related directly to the research questions. I need to know why these guidelines can make a difference or add value to elicitation of quality attributes?
- It is important to ask myself during coding what participants are looking for during elicitation? I mean what is the information they want to know? Of course, they need to know functional requirements and quality attributes, but what exactly they want to know from asking customers and users? Apparently, they focus on knowing the actors and actions of any function. They also need to know what exactly will happen to achieve quality attributes.

3- Section 3: Coding

- Sample from Participant 1

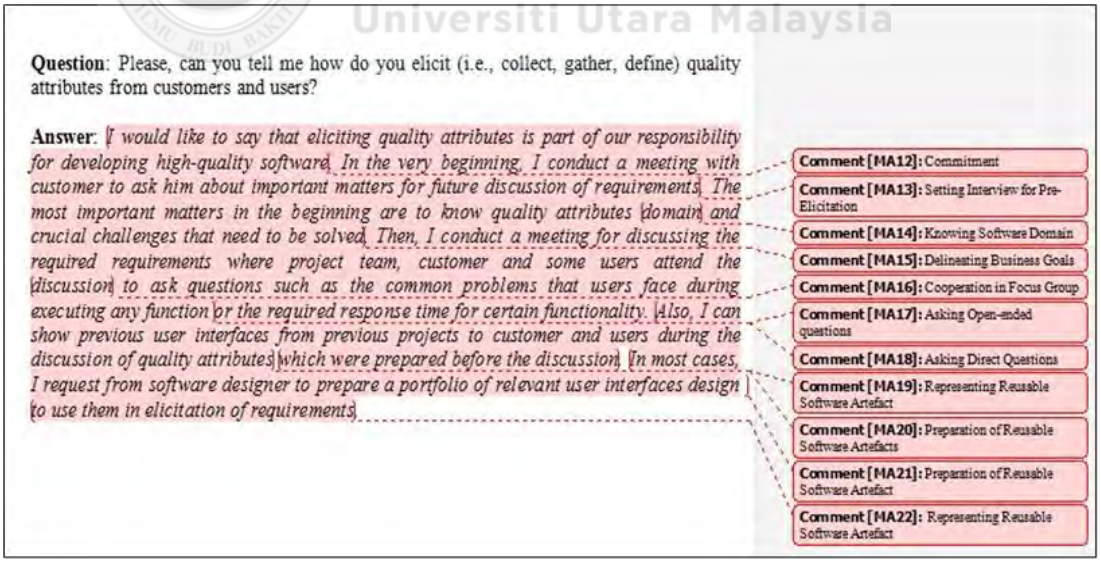

Figure D.1: Attaching Code to One Answer Of Participant 1

Commitment is a code that refers to the willing of Scrum team to do the required effort to make eliciting quality attributes successful. The quote that refers to this code is "*I would like*

to say that eliciting quality attributes is part of our responsibility for developing high-quality software".

Setting an Interview for Pre-elicitation is a code that refers to conducting an interview to know essential matters before conducting second interview with customers and users. The quote that manifests this code is *"In the very beginning, I conduct a meeting with customer to ask him about important matters for future discussion of requirements".*

Knowing Software Domain is a code that refers to refers to the subject area in which the software system is intended to apply (e.g., airline reservation, E-commerce, supply chain management). The quote that manifests this code is "*The most important matters in the beginning are to know quality attributes domain*".

Delineating Business Goals is a code that refers to underscoring the objectives of customer behind developing the software. The quote that manifests this code is "*The most important matters in the beginning are to know [...] crucial challenges that needs to be solved*".

Cooperation in a Focus Group is a code that refers to the collaboration of Scrum team members during elicitation of requirements. The quote that manifests this code is *"Then, I conduct a meeting for discussing the required requirements where project team, customer and some users attend the discussion".*

Asking open-ended Questions is a code that refers to the questions that enable customers and users to elaborate on their challenges. The quote that manifests this code is "*to ask questions such as the common problems that users face during executing any function*".

Asking Direct Questions is a code that refers to the questions that need to be answered directly to identify key elements of functions and quality attributes. The quote that manifests this code is "*or the required response time for certain functionality*".

Representing Reusable Software Artefact is a code that refers to presenting artefacts from previous projects to customer and users for compiling the details of functions and quality attributes during elicitation. The quote that manifests this code is "*Also, I can show previous user interfaces from previous projects to customer and users during the discussion of quality attributes*".

Preparation of Reusable Software Artefact is a code that refers to preparation of oft software artefacts from previous projects to customer and users for compiling the details of functions and quality attributes. The quotes that manifest this code are "*which were prepared before the discussion. In most cases, I request from software designer to prepare a portfolio of relevant user interfaces design to use them in elicitation of requirements*".

Table D.1

List of Codes

Code Name	Code Description	Quotes
▪ Commitment	▪ The willing of Scrum	▪ "*I would like to say*

	team to do the required effort to make eliciting quality attributes successful.	*that eliciting quality attributes is part of our responsibility for developing high-quality software".*
▪ Setting an Interview for Pre-Elicitation	▪ Conducting an interview to know essential matters before conducting second interview with customers and users.	▪ *"In the very beginning, I conduct a meeting with customer to ask him about important matters for future discussion of requirements".*
▪ Knowing Software Domain	▪ Knowing Software Domain is a code that refers to refers to the subject area in which the software system is intended to apply (e.g., airline reservation, E-commerce, supply chain management).	▪ "*The most important matters in the beginning are to know quality attributes domain*".
▪ Delineating Business Goals	▪ Underscoring the objectives of customer behind developing the software.	▪ "*The most important matters in the beginning are to know ... crucial challenges that need to be solved*".
▪ Cooperation in Focus Group	▪ The collaboration of Scrum team members during elicitation of requirements.	▪ *"Then, I conduct a meeting for discussing the required requirements where project team, customer and some users attend the discussion".*
▪ Asking Open-ended Questions	▪ The questions that enable customers and users to elaborate on their challenges.	▪ "*to ask questions such as the common problems that users face during executing any function*".
▪ Representing Reusable Software Artefact	▪ Presenting artefacts from previous projects to customer and users for compiling the	▪ "*Also, I can show previous user interfaces from previous projects to*

	details of functions and quality attributes during elicitation.	*customer and users during the discussion of quality attributes*".
▪ Preparation of Reusable Software Artefacts	▪ Preparation of software artefacts from previous projects to customer and users for compiling the details of functions and quality attributes.	▪ "*…which were prepared before the discussion. In most cases, I request from software designer to prepare a portfolio of relevant user interfaces design to use them in elicitation of requirements*".

- **Sample from Participant 2**

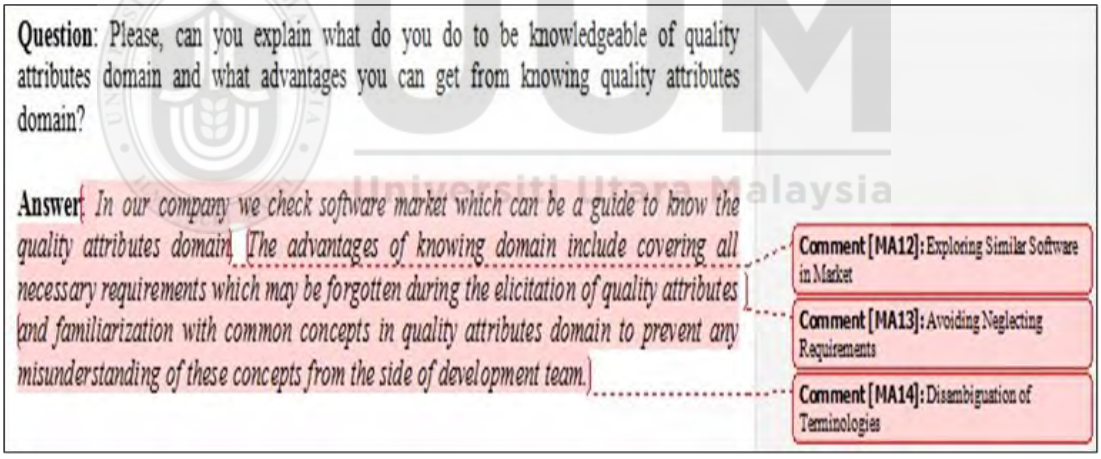

Figure D.2: Attaching codes to one answer of Participant 2

Exploring Similar Software in Market is a code that refers to investigating software systems that share the same domain of interest. The quote that refers to this code is "*In our company we check software market which can be a guide to know the quality attributes domain*".

Avoiding Neglecting Requirements is a code that refers to reminding customers and users to recall additional details about any forgotten quality attribute during the elicitation of requirements. The quote that refers to this code is "*The advantages of knowing domain*

include covering all necessary requirements which may be forgotten during the elicitation of quality attributes".

Disambiguation of Terminologies is a code that refers to understanding business jargon and associated terminologies helps to establish effective communication with customers and users and underpinning their needs in a precise manner. The quote that refers to this code is "*familiarization with common concepts in quality attributes domain to prevent any misunderstanding of these concepts from the side of development team*".

Table D.2

List of Codes

Code Name	Code Description	Quotes
▪ Exploring Similar Software in Market	▪ Investigating software systems that share the same domain of interest	▪ *"In our company we check software market which can be a guide to know the quality attributes domain"*
▪ Avoiding Neglecting Requirements	▪ Reminding customers and users to recall additional details about any forgotten quality attribute during the elicitation of requirements.	▪ *"The advantages of knowing domain include covering all necessary requirements which may be forgotten during the elicitation of quality attributes"*
▪ Disambiguation of Terminologies	▪ Understanding business jargon and associated terminologies helps to establish effective communication with customers and users and underpinning their needs in a precise manner	▪ *"familiarization with common concepts in quality attributes domain to prevent any misunderstanding of these concepts from the side of development team"*

- **Sample from Participant 3**

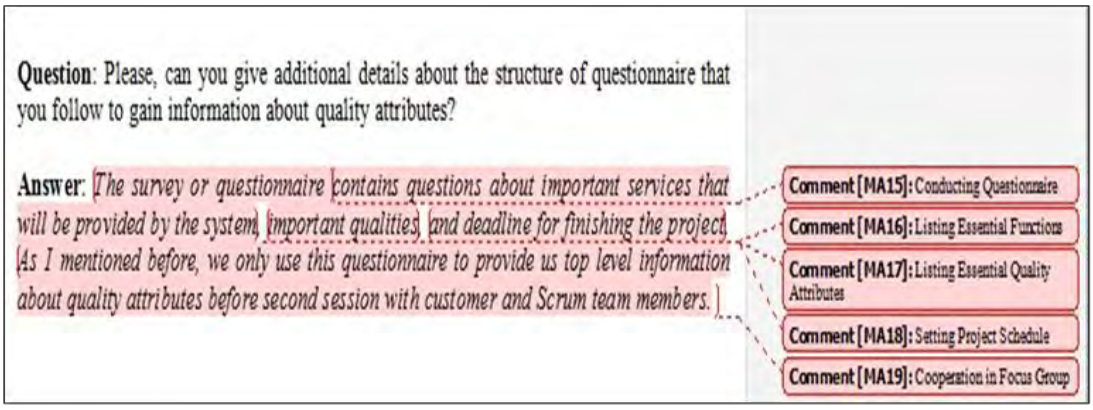

Figure D.3: Attaching codes to one answer of Participant 3

Conducting questionnaire is a code that refers to sending a set of questions to be answered by customer. The quote that refers to this code is "*The survey or questionnaire contains questions*".

Listing Essential Functions is a code that refers to the primary and basic functionalities that will be provided by software. The quote that refers to this code is "*questions about important services that will be provided by the system*".

Listing Essential Quality Attributes is a code that refers to the primary and basic quality attributes that will be provided by software. The quote that refers to this code is "*important qualities*"

Setting Project Schedule is a code that refers to knowing the deadline of finishing the project to set a timetable for finishing the project. The quote that refers to this code is "*deadline for finishing the project*".

Cooperation in Focus Group is a code refers to the collaboration of Scrum team members during elicitation of requirements. The quote that refers to this code is "*second session with customer and Scrum team members*".

Table D.3

List of Codes

Code Name	Code Description	Quotes
▪ Conducting questionnaire	▪ Sending a set of questions to be	▪ *"The survey or questionnaire*

	answered by customer.	*contains questions"*
▪ Listing Essential Functions	▪ Listing the primary and basic functionalities that will be provided by software.	▪ "*questions about important services that will be provided by the system*".
▪ Listing Essential Quality Attributes	▪ Listing the primary and basic quality attributes that will be provided by software.	▪ "*important qualities*"
▪ Setting Project Schedule	▪ Knowing the deadline of finishing the project to set a timetable for finishing the project.	▪ "*deadline for finishing the project*".
▪ Cooperation in Focus Group	▪ The collaboration of Scrum team members during elicitation of requirements.	▪ "*second session with customer and Scrum team members*".

- **Sample from Participant 4**

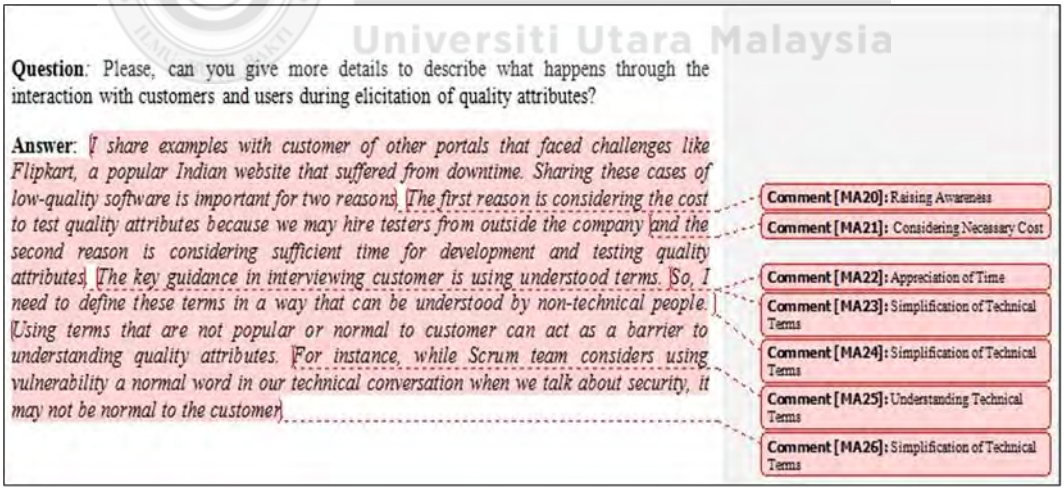

Figure D.4: *Attaching codes to one answer of Participant*

Raising Awareness is a code that refers to emphasising the importance of quality attributes and showing the possible consequences that result from neglect quality attributes. The quote that refers to this code is "*I share examples with customer of other portals that faced challenges like Flipkart, a popular Indian website that suffered from downtime*".

Considering Necessary Cost is a code that refers to taking into consideration necessary cost that maybe necessary for developing high-quality software. The quote that refers to this code is "*The first reason is considering the cost to test quality attributes because we may hire testers from outside the company*".

Appreciation of time is a code that refers to taking into consideration the required time for developing high-quality software. The quote that refers to this code is *"the second reason is considering sufficient time for development and testing quality attributes"*

Simplification of Technical terms is a code that refers to making the concepts of quality attributes less complicated by avoiding using the technical jargon and transforming the technical terminologies into definitions that are easier to grasp by customer and users. The code that refers to this code is "*The key guidance in interviewing customer is using understood terms*".

Understanding Technical Terms is a code that refers to achieving common understanding of technical terms after simplification of technical terms. The quote that refers to this code is "So, *I need to define these terms in a way that can be understood by non-technical people. Using terms that are not popular or normal to customer can act as a barrier to understanding quality attributes*".

Table D.4

List of Codes

Code Name	Code Description	Quotes
▪ Raising Awareness	▪ Emphasising the importance of quality attributes and showing the possible consequences that result from neglect quality attributes.	▪ "*I share examples with customer of other portals that faced challenges like Flipkart, a popular Indian website that suffered from downtime*".
▪ Considering Necessary Cost	▪ Taking into consideration necessary cost that	▪ "*The first reason is considering the cost to test quality*

	maybe necessary for developing high-quality software.	*attributes because we may hire testers from outside the company".*
▪ Appreciation of time	▪ Taking into consideration the required time for developing high-quality software.	▪ *"the second reason is considering sufficient time for development and testing quality attributes"*
▪ Simplification of Technical terms	▪ Making the concepts of quality attributes less complicated by avoiding using the technical jargon and transforming the technical terminologies into definitions that are easier to grasp by customer and users.	▪ "*The key guidance in interviewing customer is using understood terms".*
▪ Understanding Technical Terms	▪ Achieving common understanding of technical terms after simplification of technical terms.	▪ "So, *I need to define these terms in a way that can be understood by non-technical people. Using terms that are not popular or normal to customer can act as a barrier to understanding quality attributes*".

- **Sample from Participant 5**

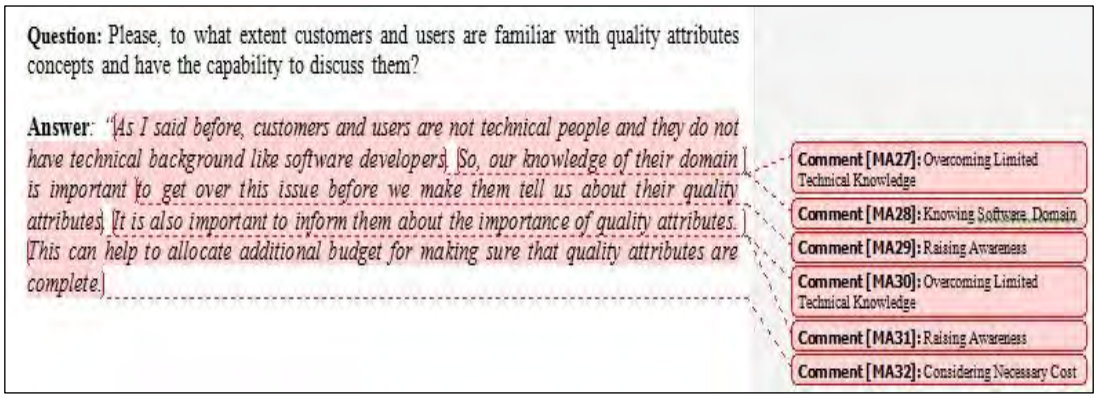

Figure D.5: Attaching codes to one answer of Participant 5

Overcoming Limited Technical Knowledge is a code that refers to overcoming the problem regarding insufficient technical knowledge of customers and users by having the knowledge of software domain to guide customers and users on technical matters. The code that refers to this code is "*As I said before, customers and users are not technical people and they do not have technical background like software developers. So, our knowledge of their domain is important to get over this issue before we make them tell us about their quality attributes*".

Raising Awareness is a code that refers to emphasising the importance of quality attributes and showing the possible consequences that result from neglect quality attributes. The code that refers to this code is *"It is also important to inform them about the importance of quality attributes"*.

Considering Necessary Cost is a code that refers to taking into consideration necessary cost that maybe necessary for developing high-quality software. The code that refers to this code is "*This can help to allocate additional budget for making sure that quality attributes are complete"*

Table D.5

List of Codes

Code Name	Code Description	Quotes
▪ Overcoming Limited	▪ Overcoming the	▪ "*As I said before,*

Technical Knowledge	problem regarding insufficient technical knowledge of customers and users by having the knowledge of software domain to guide customers and users on technical matters	*customers and users are not technical people and they do not have technical background like software developers. So, our knowledge of their domain is important to get over this issue before we make them tell us about their quality attributes".*
▪ Raising Awareness ▪ Considering Necessary Cost	▪ Emphasising the importance of quality attributes and showing the possible consequences that result from neglect quality attributes ▪ Taking into consideration necessary cost that maybe necessary for developing high-quality software	▪ *"It is also important to inform them about the importance of quality attributes".* ▪ "*This can help to allocate additional budget for making sure that quality attributes are complete"*

- **Sample from Participant 6**

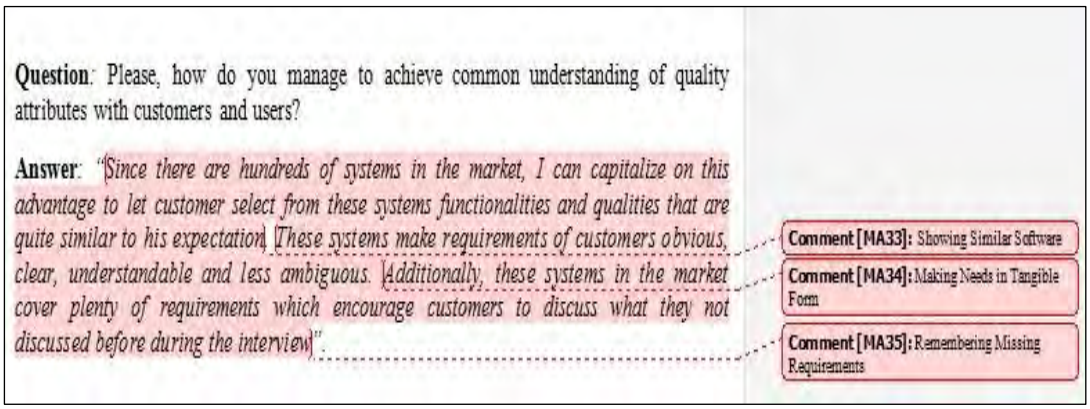

Figure D.6: Attaching codes to one answer of Participant 6

Showing Similar Software is a code that refers to presenting and enabling customers and users to see similar software from the market during requirements elicitation. The quote that refers to this code *"Since there are hundreds of systems in the market, I can capitalize on this advantage to let customer select from these systems functionalities and qualities that are quite similar to his expectation"*

Making Needs in Tangible Form is a code that refers to making requirements in visual form like user interfaces to make customers and users express their need more precisely. The quote that refers to this code is "*These systems make requirements of customers obvious, clear, understandable and less ambiguous"*

Remembering Missing Requirements is a code that refers to reminding customers and users what they did not remember during discussing requirements. The quote that refers to this code is *"Additionally, these systems in the market cover plenty of requirements which encourage customers to discuss what they not discussed before during the interview"*.

Table D.6

List of Codes

Code Name	Code Description	Quotes
▪ Showing Similar Software	▪ Presenting and enabling customers and users to see similar software from the market during requirements elicitation.	▪ *"Since there are hundreds of systems in the market, I can capitalize on this advantage to let customer select from these systems functionalities and qualities that are quite similar to his expectation"*
▪ Making Needs in Tangible Form	▪ Making requirements in visual form like user	▪ "*These systems make requirements of customers*

		interfaces to make customers and users express their need more precisely.	*obvious, clear, understandable and less ambiguous"*
▪ Remembering Missing Requirements	▪ Reminding customers and users what they did not remember during discussing requirements.		▪ *"Additionally, these systems in the market cover plenty of requirements which encourage customers to discuss what they not discussed before during the interview".*

- Sample from Participant 7

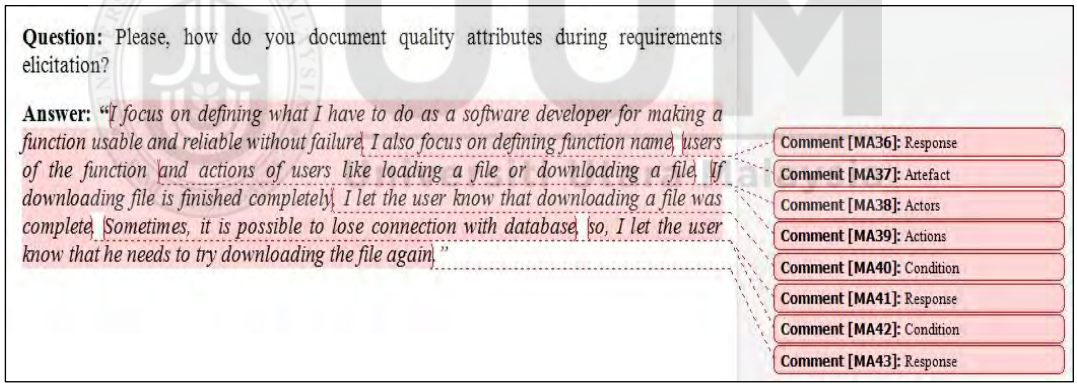

Figure D.7: Attaching codes to one answer of Participant 7

Response is a code that refers to reaction to any action performed by actors and is determined by a condition and can be in the form of showing an error message or notification of success. The quote that refers to this code is "*I focus on defining what I have to do as a software developer for making a function usable and reliable without failure*"

Artefact is a code that refers to part of software such as a user interface to which the requirement applies. The quote that refers to this code is "*defining function name*"

Actor is a code that refers to a person who interacts with software for the sake of providing inputs that can be processed by software and lead to generating outputs. The quote that refers to this code is "*users of the function*".

Action is a code that refers to the act performed by actors when they interact with part of software. The quote that refers to this code is "*actions of users like loading a file or downloading a file*".

Condition is a code that refers to an occasion that changes the response of software regarding actions done by actors that can be in the form of if statements. The quote that refers to this code is "*If downloading file is finished completely, I let the user know that downloading a file was complete*".

Table D.7

List of Codes

Code Name	Code Description	Quotes
▪ Response	▪ The reaction to any action performed by actors and is determined by a condition and can be in the form of showing an error message or notification of success.	▪ "*I focus on defining what I have to do as a software developer for making a function usable and reliable without failure*"
▪ Artefact	▪ A part of software such as a user interface to which the requirement applies.	▪ "*defining function name*"
▪ Actor	▪ A person who interacts with software for the sake of providing inputs that can be processed by software and lead to generating outputs.	▪ "*users of the function*".
▪ Action	▪ The act performed by actors when they interact with part of software.	▪ "*actions of users like loading a file or downloading a file*".

- **Sample from Participant 8**

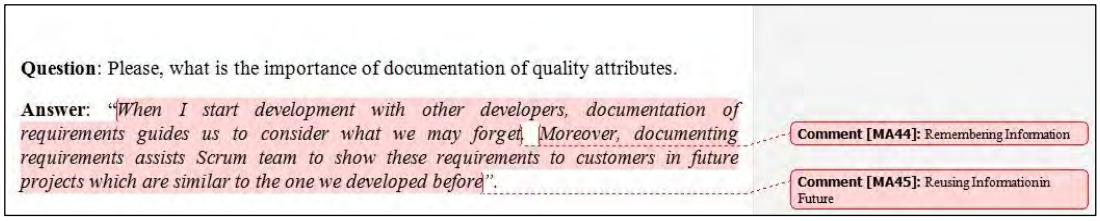

Figure D.8: Attaching codes to one answer of Participant 8

Remembering Information is a code that refers to remembering all necessary information related to key elements of functions and key elements of quality attributes during software development. The quote that refers to this code is "*When I start development with other developers, documentation of requirements guides us to consider what we may forget*.

Reusing Information in Future is a code that refers to reusing all information related to key elements of functions and key elements of quality attributes in future projects. The quote that refers to this code is *"Moreover, documenting requirements assists Scrum team to show these requirements to customers in future projects which are similar to the one we developed before"*.

Table D.8

Code Name	Code Description	Quotes
▪ Remembering Information	▪ Remembering all necessary information related to key elements of functions and key elements of quality attributes during software development.	▪ "*When I start development with other developers, documentation of requirements guides us to consider what we may forget*".
▪ Reusing Information in Future	▪ Reusing all information related to key elements of functions and key elements of quality attributes in future projects.	▪ *"Moreover, documenting requirements assists Scrum team to show these requirements to customers in future projects which are similar to the one we developed before"*.

- **Sample of Memoing during Coding**

- I decided to give a label of Commitment to answers and responses of participants that show the seriousness or exerting the required effort of participants. This code seems relevant and depicts the underlying meaning behind the responses of participants where all participants (P1, P2, P3, P4, P5, P6, P7 and P8) have shown different responses that have same meaning which is commitment.
- I decided to give a label of knowing software domain to the answers that refer to software domain that refers to the subject area in which the software system is intended to apply (e.g., airline reservation, Ecommerce, supply chain management). The responses of five participants (P1, P2, P5, P7 and P8) contributed to decide this label that seems descriptive to the exact meaning of the five participants.
- I decided to give a label of delineating business goals to the answers that refer to the main goals behind developing software. This code is descriptive to the nature of statements of participants. The responses of three participants (P1, P2 and P5) have encouraged me to select the label of delineating business goals.
- I decided to give a label of determination of target users to the answer of participant 2 that refers to the potential users of the software who are going to access the services provided by the software. While there is only one participant who is participant 2, who provided a response that influenced me to select this label, it is significant and need to be taken into consideration.
- I decided to give a label of conducting questionnaire to the answers of two participants (P3 and P6) who used questionnaire. This code is relevant because it refers to the technique of sending a set of questions to be answered by customer before setting an interview.

4- Section 4: Generating Initial Themes

- Description of Initial Theme: Proactive Exposure of Quality Attributes

The first theme which is proactive exposure to quality attributes was constructed by grouping wide array of codes which have similar characteristic which is getting elementary knowledge during and after the first contact of customers. These codes vary between participants based on their readiness and appreciation of the importance of getting this elementary knowledge in the very beginning before conducting a second interview with customers and users. These codes are: Setting Interview for Pre-Elicitation, Knowing Software Domain, Delineating Business Goals, Determination of Target Users, Defining Software Type, Reviewing Previous Projects, Exploring Similar Software in Market, Avoiding Neglecting Requirement, Disambiguation of Terminologies, Overcoming Limited Technical Knowledge, Preparation of Reusable Software Artefacts, Conducting Questionnaire, Listing Essential Functions, Listing Essential Quality Attributes, Listing Terminologies Definitions and Setting Project Schedule.

Proactive exposure to quality attributes is the first initial theme that is defined as getting elementary knowledge and insights about quality attributes during and after the first contact of customer with software organization for the sake of requesting developing software.

Getting this elementary knowledge about quality attributes is proactive in the sense that it precedes conducting a long period of discussion with customer and users about quality attributes. This elementary knowledge helps to lay the ground for successful elicitation of quality attributes from prospective customers and users.

- The Codes that construct First Initial Theme from Participants

  - Proactive exposure to quality attributes consists of six codes from Participant 1: Setting Interview for Pre-Elicitation, Knowing Software Domain, Delineating Business Goals, Reviewing Previous Projects, Disambiguation of Terminologies and Preparation of Reusable Software Artefacts.

  - Proactive exposure to quality attributes consists of nine codes from Participant 2: Setting Interview for Pre-Elicitation, Knowing Software Domain, Delineating Business Goals, Determination of Target Users, Exploring Similar Software in Market, Avoiding Neglecting Requirement, Disambiguation of Terminologies, Overcoming Limited Technical Knowledge and Preparation of Reusable Software Artefacts.

  - Proactive exposure to quality attributes consists of four codes from Participant 3: Conducting Questionnaire, Listing Essential Functions, Listing Essential Quality Attributes and Setting Project Schedule.

  - Proactive exposure to quality attributes consists of seven codes from Participant 5: Setting Interview for Pre-Elicitation, Knowing Software Domain, Delineating Business Goals, Defining Software Type, Reviewing Previous Projects, Avoiding Neglecting Requirement and Overcoming Limited Technical Knowledge.

  - Proactive exposure to quality attributes consists of five codes from Participant 6: Conducting Questionnaire, Listing Essential Functions, Listing Essential Quality Attributes, Listing Terminologies Definitions and Setting Project Schedule.

  - Proactive exposure to quality attributes consists of six codes from Participant 7: Setting Interview for Pre-Elicitation, Knowing Software Domain, Reviewing Previous Projects, Avoiding Neglecting Requirement, Disambiguation of Terminologies and Overcoming Limited Technical Knowledge.

  - Proactive exposure to quality attributes consists of five codes from Participant 8: Setting Interview for Pre-Elicitation, Knowing Software Domain, Exploring Similar Software in Market, Avoiding Neglecting Requirement, and Preparation of Reusable Software Artefacts.

Table D.9

Codes of First Initial Theme: Proactive Exposure of Quality Attributes

Codes	**Participants**

▪ Setting Interview for Pre-Elicitation	P1, P2, P5, P7, P8
▪ Knowing Software Domain	P1, P2, P5, P7, P8
▪ Delineating Business Goals	P1, P2, P5
▪ Determination of Target Users	P2
▪ Defining Software Type	P5
▪ Reviewing Previous Projects	P1, P5, P7
▪ Exploring Similar Software in the Market	P2, P8
▪ Avoiding Neglecting Requirements	P2, P5, P7, P8
▪ Disambiguation of Terminologies	P1, P2, P7
▪ Overcoming Limited Technical Knowledge	P2, P5, P7
▪ Preparation of Reusable Software Artefacts	P1, P2, P8
▪ Conducting Questionnaire	P3, P6
▪ Listing Essential Functions	P3, P6
▪ Listing Essential Quality Attributes	P3, P6
▪ Listing Terminologies Definitions	P6
▪ Setting Project Schedule	P3, P6

5- Section 5: Reviewing Themes

- Reviewing Initial Theme: Proactive Exposure of Quality Attributes

  The first initial theme (i.e. proactive exposure to quality attributes) consists of two sub-themes, namely understanding software scope and envisaging potential quality attributes. Firstly, understanding software scope is a sub-theme that is defined as knowing four essential matters, namely software domain, business goals, target users and software type from customers that Scrum team has to take into consideration when developing the software. Knowing these essential matters is a prerequisite before setting an appointment for the long discussion about quality attributes with customer and users. The quotes from five participants (P1, P2, P5, P7 and P8) contributed to develop ten primary codes that construct the sub-theme of understanding software scope. These codes are setting interview for pre-elicitation, knowing software domain, delineating business goals, determination of target users, defining software type, reviewing previous projects, exploring similar software in the market, avoiding neglecting requirements, disambiguation of terminologies and overcoming limited technical knowledge.

- The Codes of First Sub-theme: Understanding Software Scope

- Understanding Software Scope consists of five codes from Participant 1: Setting Interview for Pre-Elicitation, Knowing Software Domain, Delineating Business Goals, Reviewing Previous Projects and Disambiguation of Terminologies.

- Understanding Software Scope consists of nine codes from Participant 2: Setting Interview for Pre-Elicitation, Knowing Software Domain, Delineating Business Goals, Determination of Target Users, Reviewing Previous Projects, Exploring Similar Software in Market, Avoiding Neglecting Requirement, Disambiguation of Terminologies and Overcoming Limited Technical Knowledge.

- Understanding Software Scope consists of seven codes from Participant 5: Setting Interview for Pre-Elicitation, Knowing Software Domain, Delineating Business Goals, Defining Software Type, Reviewing Previous Projects, Avoiding Neglecting Requirement and Overcoming Limited Technical Knowledge.

- Understanding Software Scope consists of six codes from Participant 7: Setting Interview for Pre-Elicitation, Knowing Software Domain, Reviewing Previous Projects, Avoiding Neglecting Requirement, Disambiguation of Terminologies and Overcoming Limited Technical Knowledge.

- Understanding Software Scope consists of four codes from Participant 8: Setting Interview for Pre-Elicitation, Knowing Software Domain, Exploring Similar Software in Market and Avoiding Neglecting Requirement.

Table D.10:

Codes of First Sub-theme: Understanding Software Scope

Code Name	**Participants**
▪ Setting Interview for Pre-Elicitation	P1, P2, P5, P7, P8
▪ Knowing Software Domain	P1, P2, P5, P7, P8
▪ Delineating Business Goals	P1, P2, P5
▪ Determination of Target Users	P2
▪ Defining Software Type	P5
▪ Reviewing Previous Projects	P1, P5, P7
▪ Exploring Similar Software in the Market	P2, P8
▪ Avoiding Neglecting Requirements	P2, P5, P7, P8
▪ Disambiguation of Terminologies	P1, P2, P7
▪ Overcoming Limited Technical Knowledge	P2, P5, P7

Secondly, Envisaging potential quality attributes is the second-sub-theme that can be defined as expecting the potential quality attributes in the software to be developed. The data suggest that this expectation of quality attributes is not arbitrary, but based on understanding the software scope and dependent on using two techniques. These two techniques are questionnaire and software reuse that help Scrum team to make valid expectations of quality attributes. The quotes from five participants (P1, P2, P3, P6 and P8) contributed to develop six primary codes that construct the sub-theme of envisaging potential quality attributes, namely preparation of reusable software artefacts, conducting questionnaire, listing essential functions, listing essential quality attributes, listing terminologies definitions and setting project schedule.

- The Codes of Second Sub-theme: Envisaging Potential Quality Attributes

  - Envisaging Quality Attributes consists of one code from three participants (P1, P2 and P8): Preparation of Reusable Software Artefact.

- Envisaging Quality Attributes consists of four codes from Participant 3: Conducting Questionnaire, Listing Essential Functions, Listing Essential Quality Attributes and Setting Project Schedule.

- Envisaging Quality Attributes consists of five codes from Participant 6: Conducting Questionnaire, Listing Essential Functions, Listing Essential Quality Attributes, Listing Terminologies Definitions and Setting Project Schedule.

Table D.11

Codes of Second Sub-theme: Envisaging Potential Quality Attributes

Primary Code Name	Quotes from Participants
▪ Preparation of Reusable Software Artefacts	P1, P2, P8
▪ Conducting Questionnaire	P3, P6
▪ Listing Essential Functions	P3, P6
▪ Listing Essential Quality Attributes	P3, P6
▪ Listing Terminologies Definitions	P6
▪ Setting Project Schedule	P3, P6

6- Section 6: Defining and Naming Themes

1. Proactive exposure to quality attributes is the first initial theme that is defined as getting elementary knowledge and insights about quality attributes during and after the first contact of customer with software organization for the sake of requesting developing software. Getting this elementary knowledge about quality attributes is proactive in the sense that it precedes conducting a long period of discussion with customer and users about quality attributes. This elementary knowledge helps to lay the ground for successful elicitation of quality attributes from prospective customers and users.

1.1 Understanding software scope is a sub-theme that is defined as knowing four essential matters, namely software domain, business goals, target users and software type from customers that Scrum team has to take into consideration when developing the software. Knowing these essential matters is a prerequisite before setting an appointment for the long discussion about quality attributes with customer and users. This sub-theme consists of ten primary codes that construct the sub-theme of understanding software scope. These codes are setting interview for pre-elicitation, knowing software domain, delineating business goals, determination of target users, defining software type, reviewing previous projects, exploring similar software in the market, avoiding neglecting requirements, disambiguation of terminologies and overcoming limited technical knowledge.

1.2 Envisaging potential quality attributes is the second-sub-theme that can be defined as expecting the potential quality attributes in the software to be developed. The data suggest that this expectation of quality attributes is not arbitrary, but based on understanding the software scope and dependent on using two techniques. These two techniques are questionnaire and software reuse that help Scrum team to make valid expectations of quality attributes. This sub-theme consists of six primary codes that construct the sub-theme of envisaging potential quality attributes, namely preparation of reusable software artefacts, conducting questionnaire, listing essential functions, listing essential quality attributes, listing terminologies definitions and setting project schedule.

2. Mutual learning discussion is the second initial theme that is defined as discussing quality attributes with customers and users in a mutual learning fashion between customers, users and Scrum team. Mutual learning discussion means that customers and users learn from Scrum team technical matters that related to quality attributes and Scrum team learn about details of quality attributes from customers and users. This discussion is expected to be established after the customer sets an agreeable appointment for eliciting requirements which includes functional requirements and quality attributes as well.

2.1 Ameliorating technical knowledge of customers and users can be defined as increasing the level of technical knowledge of customers in terms of raising the awareness of customers regarding quality attributes importance and simplifying the concepts of quality attributes to customers and users. The aim of this sub-theme is to overcome the limited technical knowledge of customers to make them participate actively in the discussion. This sub-theme consists of nine primary codes: one-on-one discussion, cooperation in focus group, setting virtual meeting, involving users, raising awareness, appreciation of time, considering necessary cost, simplification of technical terms and understanding technical terms.

2.2 Compiling the details of quality attributes is the second sub-theme which can be defined as collecting sufficient information that is necessary to draw a comprehensive picture of quality attributes. This sub-theme consists of ten primary codes: asking open-ended questions, shedding light on challenges, asking direct questions, identifying elements of functions, identifying elements of quality attributes, representing reusable software artefacts, showing similar software, making needs in tangible form, remembering missing requirements and thinking like users.

3. Verifying common understanding is the third initial theme that is defined as examining and approving the mutual understanding of quality attributes between customers, users and Scrum team. This verification happens instantly after compiling the details of some set of requirements and can be considered as a logical extension of the second step (i.e. mutual learning discussion).

3.1 Utilization of visual artefacts is a sub-theme that is defined as employing visual artefacts that enable customers and users to see quality attributes in visual form for ensuring common

understanding of quality attributes among stakeholders. This sub-theme consists of four primary codes: Drawing Mock-up, developing proof of concept, making requirements unambiguous and deriving additional requirements.

3.2 Documentation of quality attributes is a sub-theme that is defined as writing down the details of quality attributes to enable customers and users come to an agreement with Scrum team about the quality attributes that should be included in the software. This sub-theme consists of ten primary codes: specifying artefact, specifying actors, specifying actions, specifying inputs, specifying conditions, specifying response, specifying output, specifying response measure, remembering information, creating a definition of done and reusing information in future.

Appendix F

Peer Debriefing Report

30 August 2020

TO WHOM IT MAY CONCERN

Dear Sir/Madam,

I hereby certify that I reviewed the thematic analysis that belongs to a PhD study of Mr. Hussin Ahmed Abdel-Kader Mahmoud. I found that all main procedures followed for interpretation of interviews were taken with care and in line with academic discipline. The interpretation of all quotes were delineated and supported by literature from published and coherent sources and all themes are well-explained and justified in academic way. The researcher fairly stated the themes, explained them, organized them based on their logical flow, and supported his interpretation with logic and reasonable justification. Moreover, the included figures in the writing-up document wrapped up the discussion in a clear manner. Since some quotes have shown limited technical knowledge of participants with certain aspects of quality attributes, the researcher was right to raise a flag against this limited technical knowledge for further improvement of software quality. In addition, the comparison made between the eight participants through the interpretation highlighted some differences where they collectively gave more insights of the themes investigated. In general, the comments, interpretations, and justifications are constructive and I strongly believe that the researcher was felicitous in his interpretation.

Dr. Taufiq Hail Ghilan
Certified Reviewer (Publons, Web of Science)
Researcher ID: AAV-3792-2020

Appendix G
Member Checking Report

It was a great pleasure to conduct an interview with you to explore the practices and the process of eliciting unambiguous quality attributes in Scrum-based projects. The findings of the interview resulted in three steps that construct the process of eliciting unambiguous quality attributes. Moreover, every step consists of two practices that act as guidelines for eliciting unambiguous quality attributes. Given the significance of your feedback, it will be highly valuable to review the figure that illustrates the steps, the practices and the chronology of the practices during requirements elicitation. This figure is equipped with brief definitions of the steps and the practices based on the interpretation of the interview. Moreover, the chronology is represented by a set of arrows in the figure to portray the order of following the practices during requirements elicitation. Thus, please, give your comments regarding the steps of the process of eliciting unambiguous quality attributes, the practices that help to elicit unambiguous quality attributes and the chronology of the practices during requirements elicitation. Thank you.

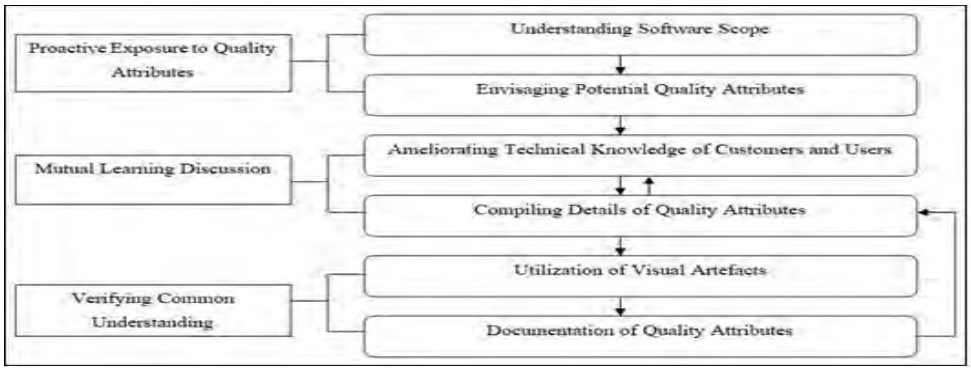

Figure F.3 The process and the practices of eliciting unambiguous quality attributes in Scrum

Table F.1

The definitions of the steps and the practices of eliciting unambiguous quality attributes in Scrum Agile method

Steps	Practices
1- Proactive exposure to quality attributes is the first step in the process of eliciting unambiguous quality attributes from customers and users. The purpose of this step is getting elementary knowledge and insights about quality attributes during the first contact of customer with software organization for the sake of requesting developing software. Getting this elementary knowledge about quality attributes is proactive in the sense that it precedes conducting a long period of discussion and interview with customer and users about quality attributes. This step consists of two practices: understanding software scope and envisaging potential quality attributes.	▪ Understanding software scope is defined as knowing the essential matters (i.e. software domain, target users, business goals and software types) from customer that Scrum team has to take into consideration when developing the software. Knowing these essential matters is a prerequisite before setting an appointment for the long discussion about quality attributes with customer and users. These essential matters are known by asking customers about them and complement that with software domain analysis via reviewing previous projects and exploring similar software in the market whenever it becomes necessary. ▪ Envisaging potential quality attributes is defined as expecting the potential quality attributes in the software to be developed. The data suggest that this expectation of quality attributes is not arbitrary, but based on understanding the software scope and dependent on using two techniques, namely conducting questionnaire and preparation of reusable software artefacts.
Mutual learning discussion is the second step in the process of eliciting unambiguous quality attributes from customers and users that is defined as discussing quality attributes with customers and users in a mutual learning fashion between customers, users and Scrum team. Mutual learning discussion means that customers and users learn from Scrum team technical matters that related to quality attributes and Scrum team learn	▪ Ameliorating the technical knowledge of customers and users can be defined as increasing the level of technical knowledge of customers in terms of raising the awareness of customers regarding quality attributes importance and simplifying the concepts of quality attributes to customers and users. ▪ Compiling the details of quality attributes is defined as collecting

about details of quality attributes from customers and users. This discussion is expected to be established after the customer sets an agreeable appointment for eliciting requirements via an interview with customer and users. The second step consists of two practices: ameliorating the technical knowledge of customers and users and compiling the details of quality attributes.	sufficient information that is necessary to draw a comprehensive picture of quality attributes. The purpose of this practice is to identify the key elements of functions and the key elements of quality attributes. Compiling the details of quality attributes is achieved via multiple techniques: asking open-ended and direct questions to customers and users, representing reusable software artefacts and showing similar software from Market and thinking like a user to suggest what can benefit customers and users.
Verifying common understanding is the third step in the process of eliciting unambiguous quality attributes that is defined as examining the mutual understanding of quality attributes between customers, users and Scrum team. This step consists of two practices: utilization of visual artefacts and documentation of quality attributes.	▪ Utilization of visual artefacts is defined as employing visual artefacts that enable customers and users to see quality attributes in visual form for ensuring common understanding of quality attributes among stakeholders. This utilization of visual artefacts is centred upon using two techniques: proof of concept and mock-ups. ▪ Documentation of quality attributes is defined as writing down the details of quality attributes to enable customers and users come to an agreement with Scrum team about the quality attributes that should be included and incorporated in the software. Documentation of quality attributes includes eight key elements of functional requirements and quality attributes: artefact, actors, actions, inputs, outputs, conditions, responses and responses measurement.

Table F.2

The Comments on the Findings: the Steps, the Practices and the Chronology of the Practices

Findings Type	**Comment**
The steps of the process	....
The practices	....
The chronology of the practices	....