# A CONCEPTUAL MODEL OF PAIR PROGRAMMING KNOWLEDGE-BASED SHARING FOR IMPROVING PROGRAMMING SKILLS

SAMARA RAHEEM ISSA

MASTER OF DEGREE
UNIVERSITI UTARA MALAYSIA
2014

# A CONCEPTUAL MODEL OF PAIR PROGRAMMING KNOWLEDGE-BASED SHARING FOR IMPROVING PROGRAMMING SKILLS

A Thesis submitted to the UUM College of Arts and Sciences in fulfillment of the requirements for the degree of Master of Science in Information Technology Universiti Utara Malaysia

by

Samara Raheem Issa

## Permission to Use

## Abstract

One of eXtream Programming practices is Pair Programming (PP) (the pair consists of a driver and a navigator), which is used for promoting knowledge sharing among students. This practice encourages students to think creatively of programming solutions, and simplify learning, especially for difficult course such as Java. By applying PP, students are enforced to improve their social skills as they communicate with each others. Despite the numerous benefits of PP (discussed by previous studies), statistics show lack of demostrating the extent at which the knowledge sharing, communication and transfer between the driver and the navigator can improve the code quality. Therefore this study aims propose a conceptual model of a PP knowledge-based sharing for improving programming skills. In order to achieve the stated objective, PP laboratory assignments were conducted and compared to evaluate the impact of PP on code quality with and without adopting the conceptual model. The conceptual model was validated by analyzing the collected data from the participants of PP laboratory assignment using Partial Least Square form of Structural Equation Modeling (PLS-SEM). The findings of the study show that socialization, combination, and internalization are the determinant factors for achieving better code quality in PP environment. The findings of this study would be benefited to academic environment especially the agile programmers in the pair programming domain.

**Keywords:** Pair programming, Tacit knowledge, Code quality and SECI model.

## Acknowledgement

I would like to express my appreciation and gratitude to everyone who has contributed in completing this thesis. I would like to express my thanks to Dr. Mazni Omar and Dr. Mazida Ahmad It was my pleasure to study under their supervision. It is not enough to thank you very much to them for their guidance to help me to achieve my goal. Without their valuable support, my thesis would not have been possible. Thanks to them for their comments which help to improve my work.

I am greatly indebted to my wonderful beloved mother Batool Saeed Ahmed for her support day by day. She has woken up whole night just to be by my side. I would like to thank my son Amir Sammer for his encouragement during my study. Thanks to my husband Sammer Saeed for his patience and understanding. My goal would not have been achieved without them.

Special thanks to my wonderful friends for their contributions and assistance. Specifically, Ms. Hanaa Khadum, Mdm. Mays Salman, Mr. Ahmed Sayed, Mr. Mohammed Fadil, Mr. Ishola, Ms. Hasana, and Ms. Wani.

I would also like to acknowledge the support and love of all of my relatives in many ways specially, Mdm. Najat Abdullah who supported me directly or indirectly in one way or another to the completion of this thesis.

I am very grateful to Dr. Azida Haji Zainol and Mam Noraziah Che Pa. They were very kind during the viva and during the period of the correction. Additionally their comments have helped to improve this work.

Finally, I am sorry for not being able to detail some of the helpful people here. I seek the magnanimity of Allah to bestow on all of them with His blessing and bountiful.
I had a very enjoyable study at Universiti Utara Malaysia (UUM). Not only, does it has a beautiful natural environment but the university also has helpful staff.

I dedicate this work as present to my father's spirit in his grave.

**Table of Contents**

## List of Tables

## List of Figures

## List of Appendices

## List of Abbreviations

| | |
|---|---|
| **CQ** | Code Quality |
| **CS** | Computer Science |
| **EK** | Explicit Knowledge |
| **KM** | Knowledge Management |
| **PLS** | Partial Least Square |
| **PP** | Pair Programming |
| **SE** | Software Engineering |
| **SECI** | Socialization Externalization Combination Internalization |
| **SEM** | Sequential Equation Modeling |
| **TK** | tacit knowledge |

# CHAPTER ONE
# INTRODUCTION

## 1.1 Overview of the Research

This study views with concern on the possibility of improving students' programming skills. Many students think that computer science (CS) and software engineering (SE) courses take the lead in dropout rates than other courses (Md Rejab, Omar, & Ahmad, 2013). This motivates the practitioners to urgently employ a technique or practice that can facilitate teaching and learning practices in CS/SE courses.

The success of Pair Programming (PP) in IT industry has been seen in terms of enhancing knowledge transfer (di Bella et al., 2013b), facilitating integration of novice members (Sillitti, Succi, & Vlasenko, 2012; SOUZA, 2012), reducing costs for training (Sillitti et al., 2012), and improving coding structure (Xu & Rajlich, 2006). This encourages practitioners in pedagogical context to rely on PP to overcome students' failure in programming course. Additionally, it not only encourages students to accept programming curricula, but also encourages to innovate in producing better end-programs (di Bella et al., 2013b). SE community accepts PP as one of many innovative approaches that had been considered to overcome programming issues in CS/SE courses (Omar, Syed-Abdullah, & Yasin,2010; Syed-Abdullah, Omar, Hamid, bt Ismail, & Jusoff, 2009). Eventually, in the late 1990s, PP has been embedded in the teaching of CS (Keefe, Sheard, & Dick, 2006; Rimington, 2010).

Improving the programming skills of the students of higher learning institutions takes the much concern of this study. Good code quality is an indicator to good programming skills. To come up with better code quality there was a need to foster the personal knowledge of the students. On that basis the idea has been initiated of constructing a conceptual model that can improve the programming skills among students of higher learning institutions which is the main aim of this study. Constructing the conceptual model came up with two needs, the first was employing a well-known model that deals with knowledge management and impacts on personal knowledge of the individuals which is tacit knowledge. The second need was using a practice that is reliable in computer science and software engineering community, deals with knowledge management and fosters tacit knowledge. For the first need this study employed the model of Nonaka and Takeuchi which is socialization-externalization combination and internalization (SECI). For the second need this study employed pair programming (PP). Pedagogically, employing PP to solve programming problems is highly related to the main concern of this study. This study has been initiated, with the conceptual (brainstorm ideas) in Figure 1.1.

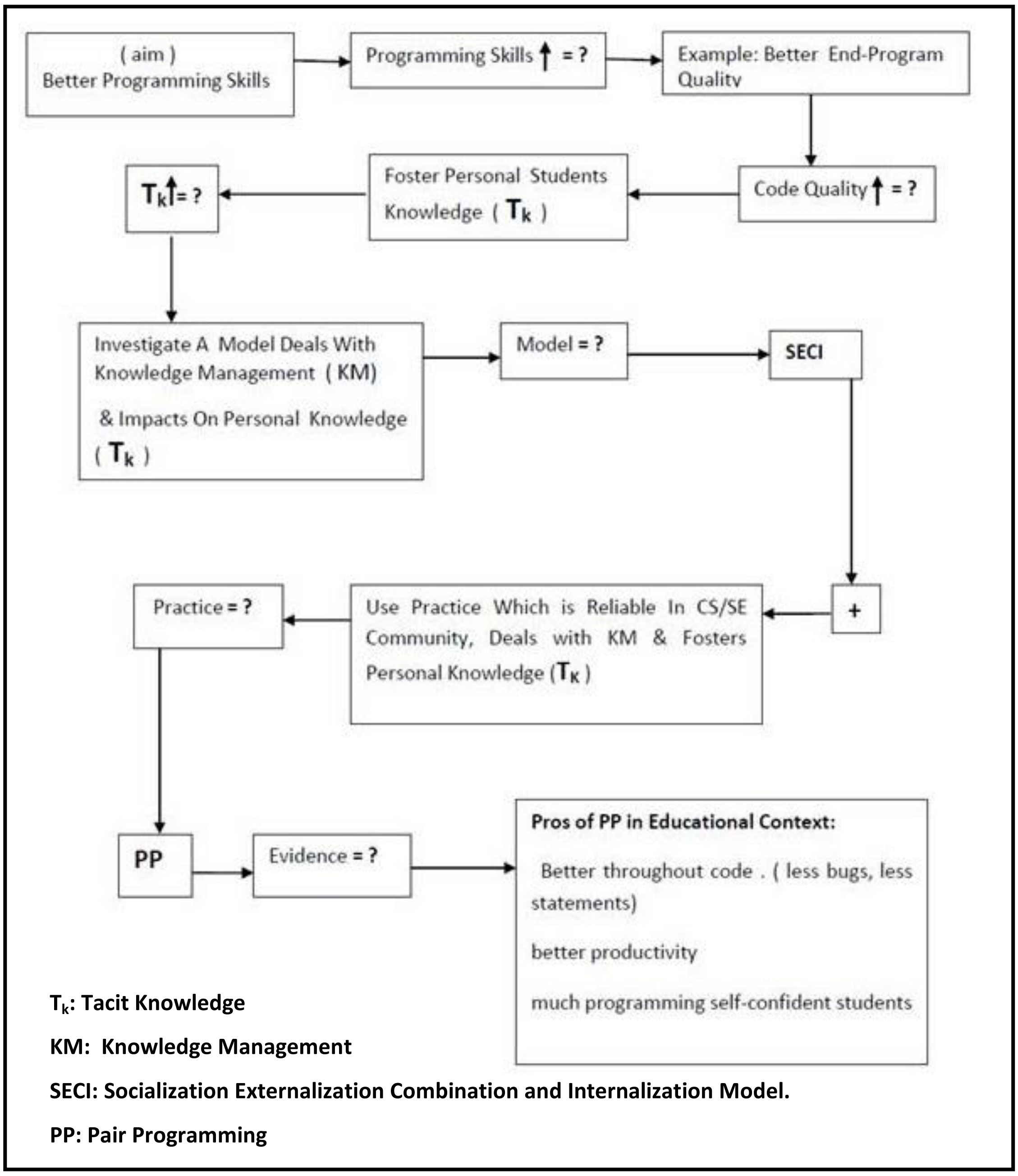
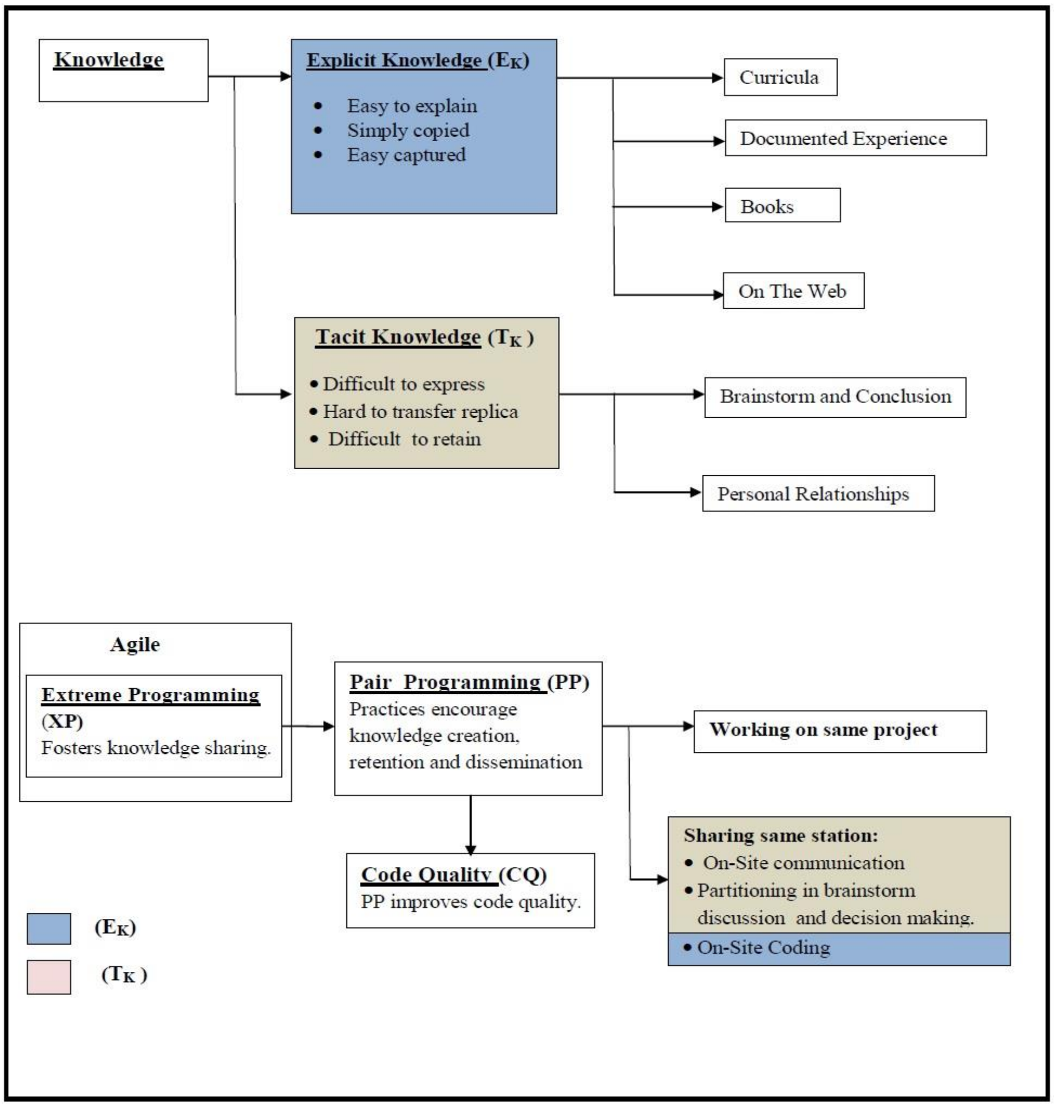

( aim ) Better Programming Skills
Programming Skills ↑ = ?
Example: Better End-Program Quality
Code Quality ↑ = ?
Foster Personal Students Knowledge ( Tk )
Tk↑= ?
Investigate A Model Deals With Knowledge Management ( KM) & Impacts On Personal Knowledge ( Tk )
Model = ?
SECI
+
Use Practice Which is Reliable In CS/SE Community, Deals with KM & Fosters Personal Knowledge (TK )
Practice = ?
PP
Evidence = ?
Pros of PP in Educational Context:
Better throughout code . ( less bugs, less statements)
better productivity
much programming self-confident students
Tk: Tacit Knowledge
KM: Knowledge Management
SECI: Socialization Externalization Combination and Internalization Model.
PP: Pair Programming

*Figure 1.1.*Conceptual idea of the study

The diagram in Figure 1.1 highlights the keywords operationally used in this study, including PP, tacit knowledge, and code quality. The keywords are concepts that are detailed out in Chapter 2. Figure 1.2 provides the relationship among those keywords on the basis of understanding the keywords and the way by which they may associate and interact. In which pair programming practices imply that pair programmers need to use same computer at same time, work on same project and discuss together till finding the solution. All of that depends on retrieving the tacit knowledge ($T_K$) of both programmers. Additionally, when pair programmers make their decision they need to write the coding that means they need to document the solution which implies transferring the tacit knowledge ($T_K$) to explicit knowledge ($E_K$) which is easy to explain and to share.



Knowledge
Explicit Knowledge ($E_K$)
• Easy to explain
• Simply copied
• Easy captured
Curricula
Documented Experience
Books
On The Web
Tacit Knowledge ($T_K$)
• Difficult to express
• Hard to transfer replica
• Difficult to retain
Brainstorm and Conclusion
Personal Relationships
Agile
Extreme Programming (XP) Fosters knowledge sharing.
Pair Programming (PP) Practices encourage knowledge creation, retention and dissemination
Working on same project
Code Quality (CQ) PP improves code quality.
Sharing same station:
• On-Site communication
• Partitioning in brainstorm discussion and decision making.
• On-Site Coding
($E_K$)
($T_K$)

*Figure 1.2*. Relationship of keywords

## 1.2 Pair Programming

SE community defines PP as one of the eXtreme Programming practices of Agile software development. It involves two persons; a driver and a navigator; who simultaneously collaborate in solving a problem using a single computer. In the collaboration, the driver types the code, while the navigator is accountable for going through the code, observes the code, and suggests alternative solutions (where necessary). Both the navigator and driver frequently switch their roles as study scheduled in advance. In short, the driver tackles operational challenges and the navigator focuses on the strategic direction by responding to the code, and recommends possible alternatives when necessary (Hannay, Arisholm, Engvik, & Sjoberg, 2010; Porter, Guzdial, McDowell, & Simon, 2013; Sajeev & Datta, 2013; Swamidurai, 2009).

## 1.3 Code quality

Code quality is an indicator for less number of defects in syntax and it measures the acceptance level of a program among users in terms of reliability, usability, maintainability, and portability (Omar, Romli, & Hussain, 2007). Besides, the literatures agree that expert opinion, effectiveness, academic performance, and number of successful test cases also measures code quality (Salleh, Mendes, & Grundy, 2011). Nevertheless, researchers also found that agile concept is a crucial factor towards achieving better software quality (de Azevedo Santos, de Souza Bermejo, de Oliveira, & Tonelli, 2011). Consequently, this relies on expert opinion to measure quality in terms of correctness criteria.

## 1.4 SECI Model

In 1995, Nonaka and Takeuchi coined a model of "knowledge creation theory", on the basis of its four-stage of activities which are socialization, externalization, combination, and internalization denoted by SECI model (Nonaka & Takeuchi, 1995). SECI modeling facilitates understanding the association of interaction and transaction between both tacit and explicit knowledge (Md Rejab, Omar, & Ahmad, 2012). Further, Ikujiro and Takeuchi (1995) detailed out the four stages. Socialization refers to a state in which tacit knowledge is generated as a result from sharing mental thinking and practical experience during social interaction like informal session, debate, and co-existence (Nonaka & Takeuchi, 1995). Externalization concerns in articulation of tacit knowledge into documents form which can be later shared with the others, based on the new codified form or explicit knowledge. Hence, externalization phase is meant by 'tacit-explicit' knowledge (Ahmad et al., 2012). Meanwhile, Combination, which is denoted by 'explicit-explicit' refers to supporting explicit knowledge with systematic resources in order to uplift the level of unsystematic explicit knowledge (Ahmad et al., 2012). Eventually, fourth phase of SECI cycle is internalization, in which systematic explicit knowledge converts to a richer, consistent, and more complicated tacit knowledge (saved in head) (Cayaba & Pablo, 2013; Jiangping, Ming, & Yahui, 2013; Santos, Goldman, & Roriz Filho, 2013). This model is embedded in the proposed model in this study, which is shown in the research model in Figure 1.3.

## 1.5 Tacit Knowledge

Tacit knowledge is a kind of personal knowledge deposited in peoples' brain as a result of expertise, imagination, learning, innovation, skills, and memory (Singh, Singh, & Sharma, 2013)**.** Educationally, during the process of knowledge transfer from an expert site to a novice site as in transferring tacit and explicit knowledge of a lecturer to students, a new tacit knowledge is to be begotten. It is worth mentioning that robust Socialization on the basis of good expert of a lecturer, strong Externalization and Combination based on various references will positively impact the Internalization of tacit knowledge in the form of skills in terms of learning, thinking, and making decision (Ahmad et al., 2012).

In 1990, Gerholm has claimed that the tacit knowledge consists of two types of knowledge; knowledge as a reflection of daily life within educational institution concerning with the process of teaching and learning administrated by the lecturer. The second is that it is begotten by students directly or indirectly as it is gained from what students have learned from the lecturer within the educational organization or knowledge that is generated from discussions and interpretation among students (Gerholm, 1990).

## 1.6 Problem Statement

This study reviews the impact of knowledge sharing in PP through latest studies. In the last two decades, PP has succeeded to attract attention of most educators, researchers, programmers, and developers. It has got a great attention in terms of its effectiveness and efficiency in reducing efforts (di Bella et al., 2013b; Porter et al., 2013).

In addition, PP has achieved a significant productivity (Salleh et al., 2011), quality improvement (Williams, Shukla, & Anton, 2004), students' interest towards Information Technology (Berenson, Williams, & Slaten, 2005; Layman, 2006; Porter et al., 2013), and self-confidence (Hannay, Arisholm, Engvik, & Sjoberg, 2010). Also, PP positively impacts students' academic performance (Salleh et al., 2011). Nevertheless, it is one of the promising and noticed knowledge sharing techniques for agile development pairs (Dybå & Dingsøyr, 2008; Kavitha & Ahmed, 2011).

However, the nature of agile development does not allow a successful knowledge sharing environment for team members because one member handles most of the tasks along PP activities over the other partners (Chigona & Pollock, 2008; Williams, McCrickard, Layman, & Hussein, 2008). It also contributes to skills differentiation, solution to conflict in time scheduling especially outside classroom framework (Williams, 2007), and non-discipline participants and absenteeism (Chigona & Pollock, 2008; Williams et al., 2008).

With reference to the descriptions in the previous paragraphs, the essential concern for this study is to overcome some of the reasons behind declination in socialization among pairs (Md Rejab et al., 2012) to ensure successful knowledge transfer between the driver and the navigator and the level of knowledge is better. The decision has been made in response to the lack of evidence in statistical data that demonstrates the extent the knowledge sharing, communication, and transfer between the driver and the navigator can improve code quality. On top of that, the decision has also been decided to take advantage of knowledge sharing platform provided by PP that contributes to issuance of better quality students of University Utara Malaysia (UUM), Collage of Art in term of Software Engineering.

## 1.7 Research Questions

Part of solving the problem addressed in the previous section, the following questions have to be answered.

1. What are the relationships between code quality and knowledge sharing in PP practice?
2. How to construct a conceptual model of pair programming knowledge-based sharing for improving programming skills?
3. How to validate the proposed model?

## 1.8 Research Objectives

This study is carried out to solve the problem discussed in the previous section and to answer the stated questions. Hence, the main aim is to propose a PP knowledge-based sharing model for improving programming skills. In order to achieve the main aim, the following objectives are formulated.

1. To determine the relationships between code quality and knowledge sharing in PP practice.
2. To construct a conceptual model of PP knowledge-based sharing for improving programming skills.
3. To validate the proposed conceptual model.

## 1.9 Scope of the Research

To meet system requirements, in educational extent, this research has been conducted at University Utara Malaysia (UUM). Particularly, first semester students of School of Computing (SOC) were involved in the empirical experiment.

## 1.10 Significance of the Study

Literatures show a lack of evidence in statistical data demonstrating the extent of knowledge sharing and communication, and that the transfer between the driver and the navigator can improve code quality (de Azevedo Santos et al., 2011). Regarding that, the results in this study enrich the empirical evidence. Therefore this study is significant to:

1. Provide empirical evidence of the relationship between the end-code quality and the knowledge sharing in PP environment.
2. Propose a conceptual model that can improve programming skills, so this model can serve as the guidelines for educators and software practitioners.

## 1.11 Conclusion of the Chapter

This chapter draws the roadmap for this study. It discusses the problem to be solved, which is complemented with research questions and objectives to be achieved. This chapter also describes the scope of study and its potential contributions. Then, Chapter 2 follows by reviewing related topics to this study and the prior similar literatures to this one. It is followed with Chapter 3 that outlines the methodology in achieving the objectives. Then Chapter 4 that goes through analyzing the data-collected and come up with the research findings. Finally Chapter 5 that discusses the findings based on the results obtained in Chapter 4.

# CHAPTER TWO
# LITERATURE REVIEW

## 2.1 Introduction

This chapter explains the concepts related to the aims of this study including Knowledge Management (KM) and its classification, modes, processes, knowledge sharing, and tacit knowledge. In addition, this chapter also describes the basic technique of PP practice and its relation to the end-code quality.

## 2.2 Knowledge

According to Yi (2006), Knowledge is a constructed information and arranged data. In 1958, Michael Polanyi has ignited the classifications of knowledge as tacit and explicit (Dorairaj, Noble, & Malik, 2012; Jiangping et al., 2013). **Tacit knowledge** is characterized by Guthrie (1996) as the own experience and expertise of a person that is hard to be described and understood by others. In addition, it is classified as the ability of calculation and decision making. It has also been defined by Li, Wang, and Cao (2013) as an applied acquisition of knowledge in a constitution that the information is not stated, which makes it very difficult to convey because it is not clearly taught or expressed. It is therefore an applied knowledge, which a person gains in doing a job everyday instead of through official instruction. This agrees with Kavitha and Ahmed (2011) who previously addressed that tacit knowledge preserves in individual’s mind in the mode of experience, memory, skills, inventiveness, and resourcefulness. This means that tacit knowledge is a resultant of an individual’s experience stored in mind, which is not easy to be formalized even not to be measured facilely and is very context-specific (Ahmad et al., 2012). Factors influencing tacit knowledge (TK) includes what the human has mentally ratified during learning phase (Billett & Choy, 2013). Besides not easy to express, it is also hard to transfer due to the differences in formulation of speech and understanding (Dudley, 2013; Ryan & O’Connor, 2013) and is difficult to retain (Argote, 2013).

In contrast, **explicit knowledge** can be transformed in a form of words, email, data (Wan, Wan, Luo, & Wan, 2011) related to tangible resources. It is supported by archived information such as curricula (Murphy & Salomone, 2013), documented experiences (Murphy & Salomone, 2013), and books in addition to web (could be a source of tacit knowledge) (Foss, Lyngsie, & Zahra, 2013). This means that explicit knowledge is easy to explain (He, Qi, Xu, & Guiyang, 2013), copy (He et al., 2013), and capture (Kimble, 2013; Okumus, 2013), and can be divulged easily (Ahmad et al., 2012).

According to Nonaka and Takeuchi (1995), there is a constant revolution of new knowledge in knowledge process, in which knowledge acqusition as well as knowledge sharing do not stop or being made to stop at any point in time. Knowledge are created and utilized through explicit and tacit knowledge interminglingly, which is refered to as knowledge conversation techniques (Koulikov, 2011). They also add that almost all the knowledge is tacit while explicit knowledge takes a small portion. It appears in Externalization stage when tacit knowledge is converted to explicit. Later, it can be reconverted back into tacit form in the Internalization stage.

## 2.3 Knowledge Management

Knowledge management has been defined by various scholars in different ways. It is seen as planning, controlling, organizing, and inspiring individuals, systems, as well as processes in an establishment so as to enhance knowledge asset and utilize it effectively (King, 2009). To some extent, it is perceived as a procedure in organizing knowledge assets in order to achieve learning in the organization (Aggestam, 2006; Alipour, Idris, & Karimi, 2011). Besides, Liao and Wu (2009) also address that knowledge management is a technique in acquiring, converting, and applying knowledge.

As the foundation, knowledge creation as well as the transformation of tacit knowledge into explicit knowledge according to Jabar, Sidi, and Selamat (2010) and Sung and Choi (2013) are recognized as the basic element of knowledge management. It is agreed by Crawford, de and Wang (2009), who found that it is achieved through person to person or cluster to cluster interaction.

Knowledge management techniques, which are ill-defined with agile methodological procedures, have been the rationale towards agile practice recognition together with software development and project. This has resulted in a wide recognition of software methodologies in various communities. According to Sharma (2014) and Singh et al. (2013), such agile techniques include PP, onsite customer, and scrum meetings also enhance knowledge creation, retention, as well as knowledge dissemination. Commonly, within any organization individuals are treated with many activities concerning knowledge involving acquiring, using, sharing in addition to sorting knowledge (Rejab, Omar, & Ahmad, 2011).

## 2.4 Knowledge Management in the Industry

In the last decades, there have being extensive works concentrating on knowledge management in organizations. It has been discovered that the capability of an organization to generate, distribute, and exemplify its knowledge in terms of services significantly determines how successful it is. Kavitha and Ahmed (2011) believe that knowledge management domain is all about enhancing the effectiveness of knowledge generation, recognition, codification, sharing, and preservation among members of an organization. This explains that high tacit knowledge should be transferred into written form, which is explicit to be readable by other persons in the organization. This is applicable to various contexts such as industry (Alegre, Sengupta, & Lapiedra, 2013; Kashefipour, Falconer, & Lin, 2002), healthcare (Abidi, Cheah, & Curran, 2005; Straus, Tetroe, & Graham, 2013; Straus et al., 2013), and library (Dalkir, 2013). Similarly, knowledge management has been perceived by Alipour et al. (2011) as a technique of organizing knowledge assets towards learning in the organization. These knowledge asset according to King (2009) can either be a document or electronic source, which include knowledge management system including a database, which is supervised by way of computer-support information and communication techniques (Cayaba & Pablo, 2013).

With regards to transferring tacit knowledge into explicit, which involves experts and novices, Nonaka and Takeuchi (1996) suggest four processes as outlined in SECI model, which are (1) Socialization (experts' tacit knowledge-novices' tacit knowledge), (2) Externalization (novices tacit knowledge-other individual's explicit knowledge) (3) Combination (individual's explicit knowledge-group explicit knowledge), and (4) Internalization (group explicit knowledge-organizational and individual's tacit knowledge). Later on, SECI's four processes were extended into six processes by Nissen (2002), which has been applied in non-organizational as well as organizational business contexts. It inspires Stankosky's (2005) four knowledge management types, which include leadership/managerial, organization, technology, learning; in non-governmental as well as governmental business contexts.

## 2.5 Knowledge Management in Education

Tacit and explicit knowledge in education describe the relationship between lecturers' and students' knowledge (Dalkir, 2013). Tacit knowledge refers to lecturers' ideas and expertise, whereas explicit knowledge is represented by lectures via computer, over the Internet, and saved in database (Liaw, Huang, & Chen, 2007). Additionally, tacit knowledge is categorized by Leonard and Insch (2005) into three categorizations; self-organization and self-motivation (referred to as cognitive), individual and institutional tasks (denoted as technical), and social (reflects social interaction).

## 2.6 Models in knowledge management

Organizations have adopted knowledge management as a strategy to gather, preserve, and analyze rich knowledge in order to gain competitive advantages over their competitors**.** The following subsections elaborate the models of knowledge management.

### 2.6.1 Wiig KM Model (1993)

It has been proposed that if knowledge management is regimented, it will be very valuable (Wiig, 2003). According to Wiig, valuable component in knowledge management models include perception, relationship, coherence, comprehensiveness, as well as objectives. Perception and objectives refer to how something is known in a specific opinion for a particular goal. Relationship explains the association among distinct objects of knowledge. Comprehensiveness is regarding how vital knowledge is offered from a particular source. However, explicit and implicit knowledge in terms of knowledge from individual remain as the spring of knowledge. In a situation where concepts, piece of information, and interpersonal connection between the entities are in accord, such knowledge are said to be coherent. Therefore, for practitioners to assume an advanced technique in knowledge source supervision, knowledge management is very vital.

### 2.6.2 Schultze and Leidner caution (2002)

The knowledge management model by Schultze and Leidner (2002) argues that knowledge and how knowledge is shared are regularly indiscriminating and importantly normative in order to degenerate and overlook the continuum. According to Kavitha and Ahmed (2011), knowledge is very much beyond an indispensable quantity of accrued data, which exist unfamiliar with the people, which are to be stored, maneuvered, and transmitted on the basis of a specific transferring device.

## 2.7 Elements of Knowledge Management

Technology, management, and organization are the elements of many knowledge management models. According to Tobin (1998), the infrastructure of Information Technology includes the repository for knowledge, which stores all the common information concerning the company and knowledge directory, which indicates where the data is stored. Other knowledge management elements are in form of systems related to experts and integrated performance.

Organizationally, knowledge management elements follow the roles of Davenporant and Ptuzak (1998), who categorize knowledge based on persons' roles within an organization. They classify roles into four groups which are: knowledge-oriented person – all people use and manage the knowledge in their daily work; knowledge management managers specialists – who manage knowledge stored in the repository; knowledge project manager – system developers who analyze the project and coordinate team members; and chief knowledge officer – serves as an advocate or evangelist for knowledge, who also involve in learning, designing, implementing, and overseeing the organization's knowledge infrastructure, and act as the primary liaison between external providers of information and knowledge.

## 2.8 Knowledge Sharing

In any organization, there is a need to administer the knowledge shared. The importance of knowledge sharing in educational context highlights when there is a need to confirm that the students are able to understand the needs analysis (Ahmad et al., 2012) and the knowledge that comes together and as their consequence.

Knowledge sharing, as maintained by agilest, is an answer to the current challenges and popular difficulties of software development. It is the main part of knowledge management and a critical mission in the processes in Agile (Rejab et al., 2011). Further, it is an organized process including investigating, selecting, reorganizing, fixing in addition to delivering the information that can improve personal skills when attending field (Sommerville & Craig, 2012). According to Crawford et al. (2013), the essence of interaction and collaboration in software development is stressed by the agile values and principles. However, collaboration which includes creative work can be expressed as interaction connecting individuals to a socio-cultural setting. Based on the literature, the knowledge lifecycle can be summarized in Figure 2.1

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Knowledge Lifecycle | = | creation (brainstorm) | + | codification (programming) | + | transfer (production) | + | application (use) |

*Figure 2.1.* Relationship of elements in related to knowledge sharing

## 2.9 System Development Methods

Dorairaj et al. (2012) address that system development is a progression of knowledge concentrated program that involves collations of needs. Examining problems, coding, inventing, assessing, as well as making sure that software are current and free from errors are among the activities. The rapid progress in Information Technology industry has increased the needs of ICT to contribute in prosperity of individuals and automation of human activities. Among many system development methodologies, Agile has emerged and has gained good worldwide reputation (Crawford et al., 2013). It encompasses recognized rules from conventional style to the development of software, which includes iterative and simple concept intended for quick development and distribution of efficient software, serious client partnership, great quality, reduced cost, as well as enthusiasm in the period of continuous changes of a project (de Azevedo Santos et al., 2011). Over the last decades, there has been a shift from conventional software development techniques, which exhibits that Agile technique has been given a highly positive reception (Crawford et al., 2013).

## 2.10 Agile

In reaction to the concerns on ideational techniques, which include waterfall technique, Agile has been a frivolous model for software development (Dingsøyr, Nerur, Balijepally, & Moe, 2012; Santos et al., 2013; SOUZA, 2012). It is because Agile meets the volatile needs of stakeholders, that sharing knowledge is a considerable way to overcome challenges in developing systems. To deliver a creative work, respond to volatility, deal with frequently changing demands of stakeholders, Agile team must be shielded with cross-functional members through collaboration of customers with developers and their interaction as well as frequent meetings. In fact, according to Crawford et al. (2013), Agile has achieved a universal recognition for its potentials in enhancing software teams' efficiency in numerous degrees by way of encouraging team's focus atmosphere, inspiring individuals, and concentrating on the clarity and outcomes of a project.

Additionally, Agile teams according to Dorairaj et al. (2012) overlap efficient teams that enhance knowledge sharing about specific project by means of physical and effective conversation as well as partnership with stakeholders. However, Agile teams usually have to struggle to be all well-informed and therefore, sharing of knowledge among team members is very essential towards the accomplishment of Agile projects.

Various studies on the implementation of eXtreme Programming techniques include the interpretation and enhancement of team communication that improves the capability of Agile project; qualitative enhancement of commitment, as well as interaction among team and compliance of experts in the developmental project. According to SOUZA (2012), in terms of interaction among mankind, coordination and cooperation in an atmosphere of self-motivated team is important.

As the popularity of eXtreme Programming has increased (Kent & Selic, 2000), the practice of PP has continued to draw much attention in the eXtreme Programming, granting PP opportunity to attract programmers' attention (Lui, Barnes, & Chan, 2010). This is because it encourages the creation of tacit knowledge among team members and encourages knowledge sharing particularly through release and iteration planning, PP, and on-site customers (Dorairaj et al., 2012).

## 2.11 Pair programming (PP)

PP is a collaborative programming manner of eXtreme Programming practices of Agile software development family. What distinguishes PP from other collaborative programming styles is the terms: "driver", "navigator" and the technique they adapt to process a task (Chigona & Pollock, 2008), who sit at the same workstation with only one set of screen, mouse, and keyboard. The two persons are imposed to design, code, diagnose, and develop a project (Berenson et al., 2005; di Bella et al., 2013b; Jo Erskine Hannay et al., 2010; Lui et al., 2010; Porter et al., 2013). In the practice, both programmers enthusiastically interact among them utilizing role-base procedure (Williams, 2007). The PP in the industry sector according to Beck (2001) has been promoted by Agile programming techniques which include eXtreme Programming. However, PP usage earlier that 1990s in the field of computer science have not been found (Rimington, 2010). The driver is one of two PP partners who code for solving the problem (di Bella et al., 2013b; Sajeev & Datta, 2013; Williams, 2010; Wray, 2010) while the navigator observes driver's job while on the keyboard. However he has strategic duties; brainstorms the whole structure, focusing on tactical errors, feeding the coding with proper alternatives (di Bella et al., 2013b; Sajeev & Datta, 2013). Although the navigator may sit for a long time with saying nothing, just observing coding process, especially when the driver is proceeding well, it does not give bad impression of misunderstanding what is going on. Both partners must remain vigilant and ready to guide the driver and pick errors up along the work (Sajeev & Datta, 2013; Wray, 2010). The roles are scheduled for a switch.

## 2.12 The way PP session is conduct

PP involves two individual programmers (the drivers and the navigators), acting as a team via similar algorithms, design, code, and test using the same computer. The driver is in charge of inputting device in order to produce the code. Meanwhile, the navigator constantly and enthusiastically assessing the work of the driver to see if there is any flaw, consider other substitute, think through strategic inferences, as well as asking questions. By so doing, recognizing the strategic paucity in the process of coding is also the role of the navigator. However, the roles as the navigator and the driver often swap, to improve their work in one way or another by practicing and learning appropriate skills when there are changes in their work routine, which occurs at the instance of natural transition during the coding pursuit (Porter et al., 2013; Sajeev & Datta, 2013; Werner et al., 2013).

### 2.12.1 PP in Academia

In academia, Williams et al. (2008) believe that PP is a good solution for preventing students from being uncertain of their programming creativity and may disappoint them or damp their interest in programming, especially when having to deal with unexpected challenges that may arise as in solo programming. In addition, PP also enforces them to improve their social skills as they communicate with each other, increase creativity sense, and starving for knowledge in IT-required businesses. Also they claim that PP is beneficial for teachers too, by damping workloads while managing less number of students' assignments as opposed to individual ones.

### 2.12.2 PP in Industry

In the industry Sajeev and Datta (2013) agree with prior studies that expert programmers show better performance when they are involved in complex missions. Since Agile methods such as eXtreme Programming began to play significant role in IT industry, novice programmers and engineers need to be familiar enough with PP aspects. In accordance, they propose striking tips. Among others, give the programmers new PP teammates, ensuring training to be familiar with shared computer set. Also, examining the possibility of using same computer with duplicated screen, keyboard, and mouse is applicable. Nevertheless, making the keyboard and mouse activated/deactivated with simple action when they swap their roles is helpful. PP is meaningful as compared with solo programming in case of complex issues. The only reason behind that new programmers and engineers are assigned with simple work under PP umbrella is to enrich team spirit, confidentiality of solving problems before proceeding to attend complex missions.

## 2.13 Agile Software Development Adoption to Knowledge Sharing

According to Cockburn (2004), the sharing of knowledge has been viewed as the main part of Agile because of its basic anticipation towards high quality and valuable software in brief statement on tacit knowledge, which are built among the teams in charge of a project through physical interactions in order to enhance competitive benefits towards the customers, as well as traverse efficient teams. Because of its brief and repetitive feature and minimal records, Boden et al. (2009) view knowledge sharing as a vital undertaking in Agile (Souza, 2012). However, Agile methods including eXtreme Programing promote collaborations and stress physical tacit sharing of knowledge within the teams and their clients or stakeholders (Beck & Andres 2004; Schwaber, 2004; Singh et al., 2013).

## 2.14 Pair Programming and Knowledge Sharing

Agile practices concern on the management of tacit Knowledge (Singh et al., 2013). In academia context, many studies have been carried out (discussed in the previous section). Further, in the context of CS, PP tactics highly support knowledge sharing (Rejab et al., 2013) towards knowledge acquisition and sharing, in which the individual relationship and association is very paramount. Though with high financial implication, PP has been acknowledged as a vital tool for knowledge sharing among members of a project team. In fact, according to Kashif and Kelly (2013), formal knowledge sharing in a workshop or project assessment meeting has also been useful in improving team members' capabilities.

Earlier, SOUZA (2012) urged that knowledge sharing is an essential technique of knowledge management that steadily changes as well as enhances the performance of an organization. In response to that, Du (2007), argued that it is connected to long-established organizational effectiveness and accomplishment, which is chained to Cummings (2004) who stressed that knowledge sharing is about solving problems, carrying out policies and techniques, acquiring new initiative through expertise collaboration, and supplying relevant information about the task to be accomplished. However, the knowledge seeker and the knowledge provider must be in an accord in order to share knowledge.

In short, Table 2.1 compares PP with knowledge sharing. When conducting a PP session, the driver and the navigator both share majority tacit knowledge and somewhat explicit knowledge (Chau & Maurer, 2004). Based on the literatures, also as mentioned in previous sections concerning ease of utilizing the explicit knowledge in terms of ease of understand, explicit knowledge can be seen in the form of numerals and words (Fengjie, Fei, & Xin, 2004; Ho, 2003; Nonaka & Konno, 1998). However, tacit knowledge is fed by opinion, expert, and think of a human (Gerard, 2003) as well as strategy of decision making (Brockmann & Anthony, 2002). As claimed by Gerholm (1990) tacit knowledge is a reflection of learning experience. Therefore tacit knowledge can be acquired along with PP practices between the driver and the navigator to produce skills of learning, thinking, and decision making (Rejab et al., 2011).

Table 2.1

*PP and Knowledge Sharing*

| **Pair Programming** | **Knowledge Sharing** |
|---|---|
| Two partners: navigator and driver | Two partners: contributor and receiver |
| Navigator: participates with the driver by addressing alternative ideas in attempts to solve the problem. The best alternative will be selected. | Contributor: subscribes some of his expertise and sends it to the receiver, who will add it to his own knowledge base. |
| Driver: writes codes brainstormed with the navigator. | Receiver: guided by the contributor. |
| Partners share place and computer to solve problems unless they use distributed PP (no time and place limitations). | Contributor and receiver communicate over knowledge sharing space (e.g., .net meeting, team viewer, yahoo messenger) |

## 2.15 Correctness of Code Quality

Previous study has stressed that quality attributes are the main determinants that impact the run-time behavior, system design, and the user experience. The contents of programming quality attributes could be recognized as conceptual integrity, maintainability, reusability, availability, interoperability, manageability and performance. Others are reliability, scalability, security, supportability, testability, and usability. Mahdavi-Hezavehi, Galster, and Avgeriou (2013) mention that the importance of quality attribute cannot be over emphasized as it has great impact on the characteristics that affect the quality of the software systems. This is a result of descriptions that quality attributes give to the system in order to balance some specified requirements.

In addition, quality attributes are described as guide that merges different quality and frameworks like IEEE standard for a software quality metric methodology or ISO standards (Abran, Moore, Bourque, Dupuis, & Tripp, 2004). In other words, the guideline, which is sometimes referred to as SWEBOK characterizes quality attributes as discernible, which are performance, security, and availability. On the other hand, quality attributes that are not discernible at runtime are recognized as modifiability, portability, and reusability. In some cases, quality attributes are found related to architecture's intrinsic qualities as conceptual integrity and correctness (Abran et al., 2004; Bass, Clements, & Kazman, 2003).

Additionally, program's code correctness has been the main issue in analyzing and working on the issues affecting the code quality in programming environment (Markoski, Hotomski, Malbaški, & Obradović, 2013). Researchers have argued that correctness of the codes could be achieved through the means of manual design and manual confirmation, which are not similar with automatic confirmation approach (Dijkstra, 1993; Iranzo, 2002). Also, one of the issues in code correctness characteristics is the compiler's correctness itself, which is a step-forward for achieving software correctness and reliability. This is as a result of current software development that is written in higher programming languages and requires translation in to machine code (Blech & Glesner, 2004). Further, Blech and Glesner (2004) stressed that correctness of codes or programs requires a formal semantics of the main programming languages, which could be intermediately represented based on the focused language. Besides that, researchers have however argued that assumption of testing the input data value should be considered heavier than the produced output in related to the correctness of code characteristics (rey Voas, 1996). Consequently, correctness of code or program should embed aspects including safety of code, failure tolerance, and testing of vulnerability in the characteristics of code correctness.

## 2.16 Code Quality and Pair Programming

One of the outcomes of PP, as claimed by Cockburn and Williams (2000) and Wood and Kleb (2003) is shorter length of code. According to Ciolkowski and Schlemmer (2002) and Cockburn and Williams (2000) shorter code is an indication to improvement in underlying design. Besides, PP reduces rate of defect (di Bella et al., 2013a; Jensen, 2003; Tomayko, 2002; Williams, 2001). They also agree that a large number of successful test cases were achieved when utilizing PP.

## 2.17 Conclusion

This chapter reviews the underlying concepts of this study. The way the concepts were utilized in the previous works are discussed at length, giving ideas to this study.

# CHAPTER THREE
# RESEARCH METHODOLOGY

## 3.1 Introduction

This chapter describes the methodology used to achieve the objectives outlined in Chapter 1. It explains the type of research that has been conducted, the sampling, and methods for collecting data.  Statistical techniques are used to analyze the data, hence they are explained briefly.

## 3.2 Research Design

The nature of this study is quantitative, which is based on a positivistic approach and is explained in terms of variables, hypotheses, and units of analysis (Neuman, 1997). This study aims to test the suitability of the proposed model based on SECI model in improving programming skills among students in higher learning institutions. Therefore, to achieve the main aim, this study employs experimentation in order to test the effectiveness of the SECI model.  While the SECI model is based on Nonaka and Takeuchi (1995), the experimentation process is adapted from Wohlin et al. (2012)  In short, the relationships of the goals, questions, and objectives of this study are outlined in Table 3.1.

Table 3.1

*Relationships of Goals, Research Questions, and Research Objectives*

| | Goal | Research Question | Research Objectives |
|---|---|---|---|
| 1 | Investigate the relationships between code quality and knowledge sharing in PP practice. | What are the relationships between code quality and knowledge sharing in PP practice? | To investigate the relationships between code quality and knowledge sharing in PP practice. |
| 2 | Construct a conceptual model of pair programming knowledge-based sharing for improving programming skills. | How to construct a conceptual model of pair programming knowledge-based sharing for improving programming skills? | To construct a conceptual model of pair programming knowledge-based sharing for improving programming skills. |
| 3 | Validate the proposed model. | How to validate the proposed model? | To validate the proposed model. |

Table 3.2 highlights how the three objectives of this study have been achieved in related to the stages of the methodology followed. In addition, the stages involved are shown in Figure 3.1.

Table 3.2

*Relationships of Objectives, Stages and Deliverable*

| Research objectives | | Phase | Deliverable |
|---|---|---|---|
| 1 | To investigate the relationships between code quality and knowledge sharing in PP practice. | Preliminary Study | Empirical evidence relationships between code quality and knowledge sharing in PP practice. |
| | | SECI Model and Quality | |
| 2 | To construct a conceptual model of pair programming knowledge-based sharing for improving programming skills. | Experimental process | Conceptual model of pair programming knowledge-based sharing for improving programming skills. |
| | | Experiment planning | |
| 3 | To validate the proposed model. | Experimental operation | Validated model. |
| | | Analysis and Interpretation Using SEM Approach | |
| | | Discussion and conclusion | |

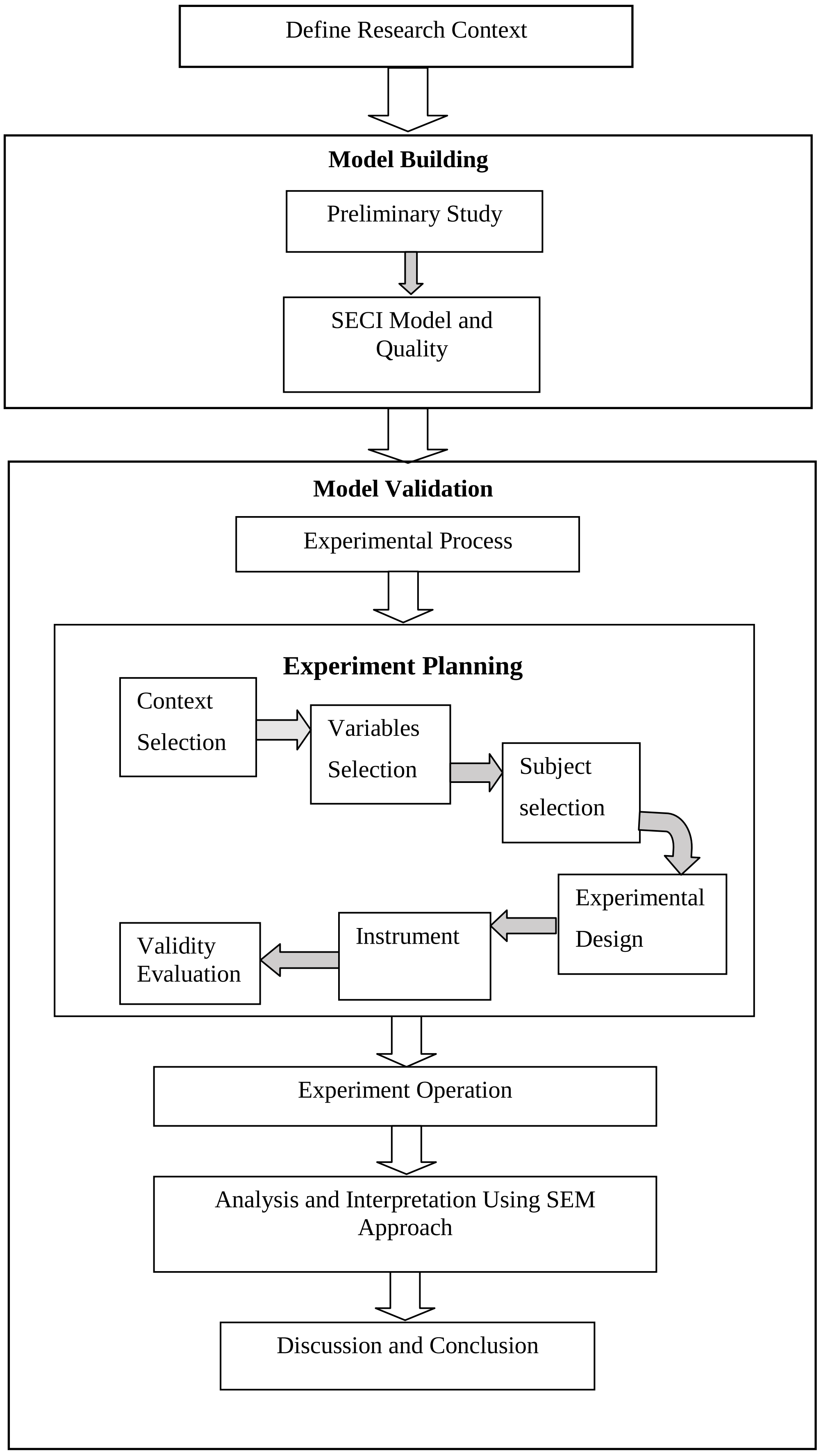


*Figure 3.1.* Research Process

## 3.3 Define Research Context

Initially, this stage defines the problem, reviews the literatures to understand the concepts, theories, and to discover previous studies related to PP practices, knowledge sharing, and the quality of program.  The readings were made on online references including from journals, books, and related papers.  Eventually, sufficient knowledge has been acquired. The acquired knowledge was helpful to identify the problem, defining the scope, and clarifying the objectives of this study.

## 3.4 Preliminary Study

A preliminary study was carried out by using questionnaires. The questionnaire in Appendix A is adapted from Rejab et al. (2013) and was distributed to students enrolled in  PP  lab experimentation and were gaining experiencing in fundamentals of java programming language. Having collected the data, the correlation was determined between knowledge sharing and PP in educational extent and a comparison was made between two PP models, in which SECI model has been adopted by one of the PP models. The results from this stage reveal influencing factors of both, (1) students' performance based on the quality of the program and (2) knowledge sharing by using SECI model in PP practices.

## 3.5 Experiment Definition

Experiment definition is the first stage of experimentation process, which was adapted from the experimentation in software engineering (Wohlin et al., 2012).  The stage identified the experiment goals based on the research questions.

With reference to Table 3.1, only the first and the second goals were involved in the experimentation, which aimed to investigate the relationships between knowledge sharing and quality of program in PP practice. Once the experimentation goals have been decided, the next stage (experiment planning) was conducted as discussed in the next section.

## 3.6 Experimental planning

Several activities were included in this stage, as outlined in the following sub-sections.

### 3.6.1 Context Selection

Context Selection is the first experiment planning activities. Undergraduate students were chosen to be the subjects for the experiment because experimenting with students is controllable and easy to replicate. They were required to apply two different PP assignments in four labs; two labs for each of the assignment, in which the level of difficulty was ensured somewhat similar. The roles of driver and navigator were swapped once for each assignment. Besides, SECI model was applied during the second assignment. This is important to suit the aim of the study, which is to validate the proposed conceptual model of PP knowledge-based sharing for improving programming skills. Additionally, this study proposes to use SECI model for knowledge transfer between the driver and the navigator in PP practice as shown in Figure 1.3.

### 3.6.2 Hypothesis Construction

The hypotheses were derived and formulated based on the experimentation definition as clarified in Section 3.5. Each hypothesis is defined in Table 3.3.

Table 3.3

*Hypotheses Of The Study*

| NO | Hypothesis | Codification | Description |
|---|---|---|---|
| 1 | H1 | $S \xrightarrow{NS} CQ$ | The Socialization process contributes positively to student's code quality without employing SECI process. |
| 2 | H2 | $E \xrightarrow{NS} CQ$ | The Externalization process contributes positively to student's code quality without employing SECI process. |
| 3 | H3 | $C \xrightarrow{NS} CQ$ | The Combination process contributes positively to student's code quality without employing SECI process. |
| 4 | H4 | $I \xrightarrow{NS} CQ$ | The Internalization process contributes positively to student's code quality without employing SECI process. |
| 5 | H5 | $S \xrightarrow{YS} CQ$ | The Socialization process contributes positively to student's code quality with employing SECI process. |
| 6 | H6 | $E \xrightarrow{YS} CQ$ | The Externalization process contributes positively to student's code quality with employing SECI process. |
| 7 | H7 | $C \xrightarrow{YS} CQ$ | The Combination process contributes positively to student's code quality with employing SECI process. |
| 8 | H8 | $I \xrightarrow{YS} CQ$ | The Internalization process contributes positively to student's code quality with employing SECI process. |
| 9 | H9 | $SECI \rightarrow CQ$ | SECI process contributes positively to student's code quality. |

### 3.6.3 Variables Selection

All variables were specified before conducting the experiment. It was helpful in overcoming the validity threats (Erdogmus, 2005). Figure 3.2 shows a legend for NSNR, NSYR, YSNR, YSYR for the independent and dependent variables involved in this study.

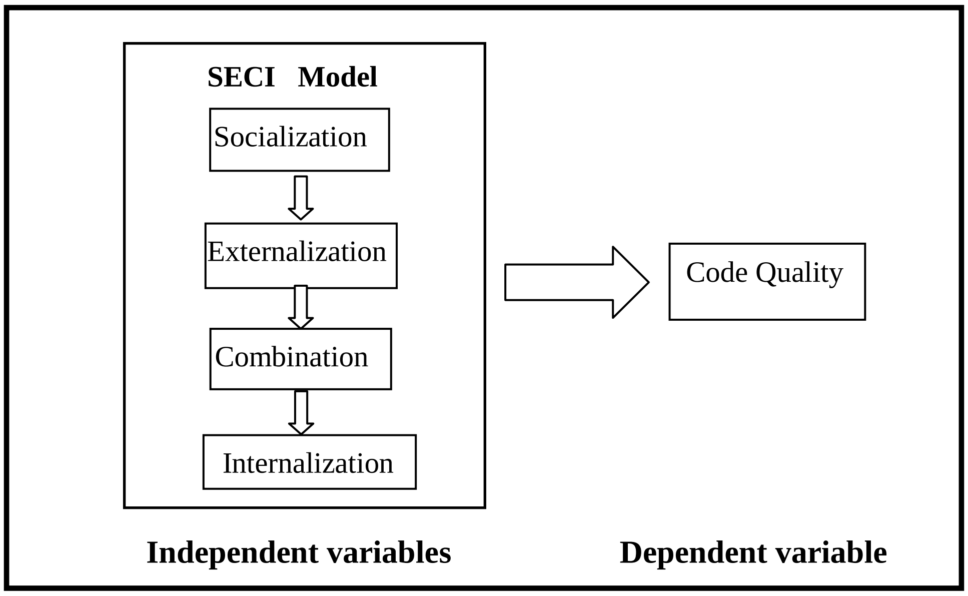


*Figure 3.2.* Independent and dependant variables

The independent variables in Figure 3.2 refer to the purpose of this study, which is in a form of PP assignments. To evaluate the quality with SECI model, the independent variables undergo the experimentation process. Meanwhile the dependent variables refer to the effects to be measured. The dependent variables are code quality and elements in the SECI model, which are Socialization, Externalization, Combination and Internalization.

### 3.6.4 Subjects Selection

This study utilizes non probability sampling, particularly purposive sampling. Purposive sampling is a technique to pick up a sample based on the needs of the study (Cohen, Manion, & Morrison, 2000). In addition, it is based on a consideration in gaining a representative sample of the people (Lavrakas, 2008). On that basis, undergraduate students of Collage of Arts and Sciences (CAS) at Universiti Utara Malaysia (UUM) were selected to involve in the experiments. The learning zone (UUM's learning management system) has been used to announce the call for participation. This ensures that all participants involved in the study on voluntary basis. They were motivated by given SIRA marks (marks for participating in co-curriculum activities). Altogether, 108 students participated who were familiar with the fundamentals of Java programming.

They were required to solve two Java programming assignments, assigned by the lecturer. They reflect students' performance in PP practices through pre and post applying SECI phases. Additionally, the participants were required to answer a set of questionnaire that reflects their perception on knowledge sharing between pair programmers in presence of SECI model.

### 3.6.5 Experiment Design

This study concerns on testing the knowledge sharing through applying SECI model and its relationship with the quality of the end-program. In addition, this study has an intention in the manipulation of variables. Hence, the decision in conducting experiment using repeated measures were taken. This makes every student involved in different situations in the experiments.

Four conditions of programming practices were included in this study, which are 1) PP without applying SECI for knowledge transfer in cases of no rotation among pairs denoted by NSNR and 2) with rotation denoted by NSYR. Also, 3) PP with SECI model in cases of no pair rotation denoted by YSNR and 4) with rotation denoted by YSYR. In short, the repeated measure design is shown in Table 3.4.

Table 3.4

*Repeated Measure Design*

| **IV \ V** | **Dependent Variable (Code Quality)** | | | |
|---|---|---|---|---|
| | **NSNR Student Group** | **NSYR Student Group** | **YSNR Student Group** | **YSYR Student Group** |
| **Independent Variables ( SECI – CQ )** | Socialization<br>Externalization<br>Combination<br>Internalization | Socialization<br>Externalization<br>Combination<br>Internalization | Socialization<br>Externalization<br>Combination<br>Internalization | Socialization<br>Externalization<br>Combination<br>Internalization |

Table 3.4 explains that every student pairs applied four different programming practices at different times. This ensures the reliability of the gathered results. The design is further detailed in Figure 3.3.

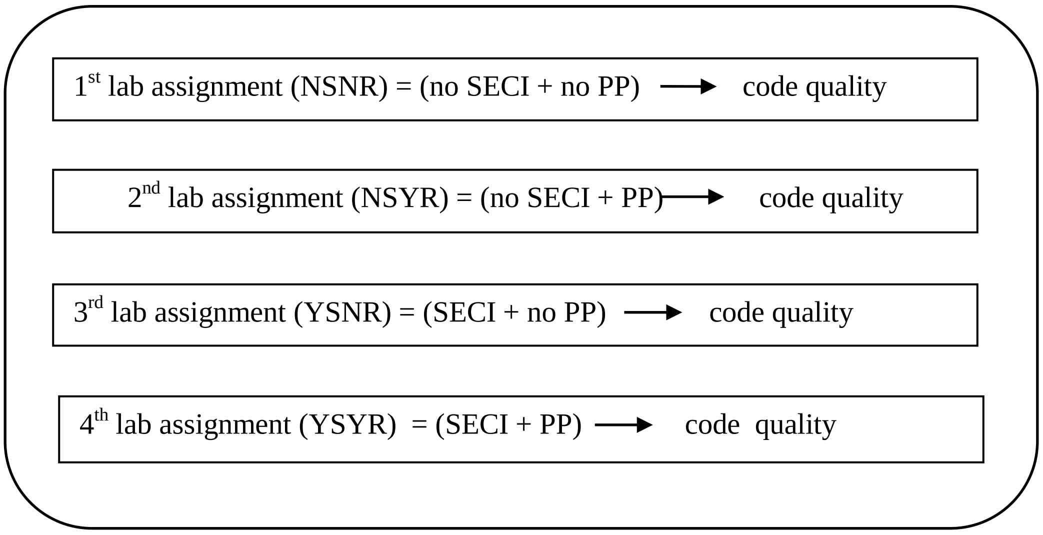

1st lab assignment (NSNR) = (no SECI + no PP) → code quality
2nd lab assignment (NSYR) = (no SECI + PP) → code quality
3rd lab assignment (YSNR) = (SECI + no PP) → code quality
4th lab assignment (YSYR) = (SECI + PP) → code quality

*Figure 3.3. Lab experiments*

Referring to Figure 3.3, the first and the second lab experimentations were concerned with PP practice with the absence of SECI implication for knowledge sharing. Meanwhile the third and fourth lab experimentations were incorporated with SECI model. Same questionnaire was distributed to the participants to measure their level of knowledge during lab activities. For the purpose of conducting effective lab experiments and to reduce the effects of biasness, several procedures were taken as detailed out in the next sections. Further, to ensure proper PP sessions in terms of interaction and collaboration, the roles of the instructor and the participants were specified in the following guidelines:

Roles of instructor:

1. Brief students on PP and its practices.
2. Give students chance to choose their adequate pair programmer.
3. Support novice participants with tips in case of difficulty to encourage them to proceed well in completing the task.

4. Explain the problems to the participants in some ways without highlighting the answer except for novice participants who could be supported with tips especially in the early stages.
5. Trace the deployment equality in participation between the pair programmers.

Roles of the participants:

1. Free to choose their adequate pair partner.
2. Ask the instructor for guidance in case of necessary.
3. Discuss with the partner to come out with proper results.
4. Switch the roles (in pairs) as scheduled.

### a. First Lab Procedure

In the first lab (the NSNR), after the participants were briefed on the concepts and practices of PP, they were free to select their partner. The pairs were given one hour to solve an assignment in Java programming, in which the sufficiency of time to solve the assignment was highly ensured. The pairs were not scheduled to swap the members' roles during this lab, implicating that each member of the pairs was either a driver or a navigator until the end of the assignment.  The session was monitored closely by a group of lecturers and the researcher.  After participants finished the task, they were given another programming question consisted of three sub tasks.

### b. Second Lab Procedure

The second lab (the NSYR) was also given one hour, but with roles rotation between the members of the pairs. This enables each member to be a driver for half an hour, while as a navigator for another half an hour. Similarly, the set of questionnaire in the first lab was distributed in this second lab too.

Meanwhile, the third and fourth labs were conducted to investigate the quality of the program in the presence of SECI model in PP practice. The equality in terms of the level of difficulty of the assignments in the four lab sessions was highly ensured. For better implementation of SECI processes, the participants were instructed with a set of guidelines (Table 3.5) before conducting the third lab. This might positively affect the knowledge sharing between the diver and the navigator of the pairs, and accordingly might impact on final program coding.

Table 3.5

*SECI Guidelines*

| SECI Stage | Guideline |
|---|---|
| Socialization | Each participant has to think in depth to come up with a program that solves the problem. |
| Externalization | The members of the pairs need to share by writing a draft code of the program. |
| Combination | The participants can refer to the Internet, software book, or any source to support their programming. |
| Internalization | Once participants are satisfied with the output code, they can write and run it using the provided computer. |

**c. Third Lab Procedure**

The third lab (YSNR) was also run in one hour. It was conducted to investigate the quality of the program with the presence of SECI and without pair rotation within PP environment. Participants were required to keep the roles of a driver or a navigator until the end of the session. After the participants finished the assignment in the third lab session, a set of questionnaire was distributed before proceeding with the fourth lab session.

**d. Fourth Lab Procedure**

Similarly, one hour was assigned for the fourth lab (YSNR). The aim was to investigate the quality of the program with the presence of SECI, and with pair rotation as illustrated in Figure 3.3. The members of the pairs were required to switch the roles as a driver and a navigator after the first 30 minutes.

### 3.6.6 Instrumentation

The use of instrument provides a mean to perform and monitor the experiment. In this study, a questionnaire was used. The questionnaire was adapted from SECI model in educational context, which includes Socialization, Externalization, Combination, and Internalization dimensions (Mazida, 2010). The questionnaire distributed at the end of the second lab session was also distributed to the participants before solving the assignment of the fourth lab session.

The instrument contains 31 items, asking participants' perception towards teaching materials, students, lecturers, and the resultant, which refer to independent of learning, independent of thinking, and independent of decision making. All items in the questionnaire were measured using a five (5) point Likert scale. In detail, 1 refers to *strongly disagree,* 2 refers to *disagree,* 3 refers to *do not know,* 4 refers to *agree,* and 5 refers to *strongly agree.* Items were constructed based on knowledge management criteria by Anantatmula and Kanungo (2005), elements of pedagogical learning environment (Ally, 2004), and factors defining the efficiency of learning programming in coordination with the process of knowledge management (Nonaka & Takeuchi, 1995). Experts have reviewed the content validity.  Besides, the reliability of the instrument is high, with Cronbach Alpha 0.91. Instrument is available in Appendix A. In order to validate the model five validities have been conducted which are validity evaluation, internal validity, external validity, construct validity and conclusion validity.

### 3.6.7 Validity Evaluation

Several threads of validity should be taken into consideration when conducting experiment and needed to be reduced. For this study, the threats of validity include internal, external, construct, and conclusion validities. They are discussed in the following sub-sections.

#### 3.6.7.1 Internal validity

This threat is a measure to what extent the participants conform to programming practices used.  There is a possibility that the participants do not follow the practice properly. Thus, they were briefed on the concepts and practices related, guidelines, and examples have been distributed to them. In addition, a group of lecturers and the researcher monitored them to ensure they conform to the assigned programming practice.

#### 3.6.7.2 External validity

External validity concentrates on whether claims for the generality of the results were justified depending on the nature of the sampling involved in the study. This study employs students as the participants. On the basis of the differences in programming settings, experience and software development tools in industrial framework, generalizing the findings to industry may be impossible. Thus, this study can be generalized based on the students, who have basic programming background.

#### 3.6.7.3 Construct validity

The correct interpretation and the right measuring for the theoretical construct take much interest of the construct validity. Therefore, to ensure the selected variables reflect the construct of the cause and effects well for this study, the researcher requested the lecturers to give feedback of the Java programming assignments submitted by the participants to assess the performance of the participants in term of the quality of programming assignment.

#### 3.6.7.4 Conclusion validity

Conclusion validity focuses on whether other researchers will be able to replicate and gain similar outcome in case of following the procedures as the original study when conducting an experiment. For that purpose, the experimental procedure was documented and detailed out in the current study precisely. Perfectly, it is not possible to eliminate the whole threats while conducting an experiment. As an example, while conducting a lab session for the experiment, in order to minimize threats to internal validity, many exotic variables are needed be controlled. In fact, it is nearly difficult to control the variables, while conducting a field experiment, despite that this experiment may increase the external validity of the study. Thus, the experiment is needed to be carefully designed for the sake of minimizing the threats to achieve validity.

## 3.7 Operation

The experiment was conducted after carefully planned. In this stage, all the procedures, training materials, and instruments were prepared and carried out. A preliminary study for the current research has been conducted as discussed in Ahmad et al. (2012).

## 3.8 Analysis and Interpretation

To meet the research objectives, quantitative analysis was used. For the purpose of testing the determined hypotheses, Structural Equation Modeling (SEM), which is an analytical technique involves measurement errors to understand the influencing indicators (Kline, 2005) was run. Also SEM was used to examine whether the conceptual model fits with the collected data through the experiments.

In this study, Partial Least Square (PLS) is employed, utilizing SmartPLS 2.0 as the tool. This is because PLS can be used to avert the limitations of co-variance-based SEM with regards to distributional properties, measurement level, sample size, model complexity, identification, and factor interdependencies (Chin, 1998; Fornell & Bookstein, 1982). Urbach and Ahlemann (2010) stated the criteria for choosing PLS, i.e. PLS makes fewer demands regarding the sample size than other methods and the input data is not necessarily normally distributed data. Also, it can be applied to a complex SEM with large number of constructs and is able to handle both reflective and formative constructs. Covariance based SEM like AMOS is used to test or confirm on the existing theory or model, but PLS can be used for theory confirmation or theory development, which includes using to develop prepositions by exploring the relationship between variables (Chin, 1998). Since the model in this study is conceptualized based on literature review, then PLS is applicable.

## 3.9 Summary of the chapter

This chapter details out the research methodology that has been carried out from the early stage of the study. It begins with research definition, followed with experiments, the instrumentation, and then going through the process of model validation.

# CHAPTER FOUR
# RESEARCH FINDINGS

## 4.1 Introduction

This chapter presents the results and analysis of data using the International Business Management (IBM) Statistical Package for Social Sciences (SPSS) version 20 and the SmartPLS 2.0 tools. The chapter commences with a description of the analysis that has to do with profile of the participants. Then, the chapter discusses the analysis of measurement model and structural model, which is specifically focusing on the formulated hypotheses.

## 4.2 Descriptive Statistics of Respondents for Experiments 1 and 2 (NSYR & YSYR)

The statistical frequency distribution of variables in the questionnaire was classified and presented in a way to reflect the originality of this study. As discussed in Chapter 3, NSYR is a denotation to the experiment in which PP session was conducted without applying SECI process but with rotation in the roles of the members of the pairs. Meanwhile, YSYS denotes the experiment with the incorporation of SECI and pair rotation.

Based on that, the descriptive analytical tables for experiments 1 (NSYR) and 2 (YSYR) were exhibited for proper data analysis and further analysis as shown in Tables 4.1 and 4.2. In experiment 1, Table 4.1 shows that 27 (25%) of the participants were male and 81 (75%) were female (from the total 108 participants).

13 (12%) of them aged between 18 and 20 years, while 86 (79.6%) were between 21 and 23 years old. The remaining 9 (8.3%) were between 24 and 26 years old.

Table 4.1

*Demographic Statistics of Experiment 1 (NSYR)*

| **Variables / Factors** | **Frequency** | **Percentage** |
|---|---|---|
| **Gender** | | |
| Male | 27 | 25 |
| Female | 81 | 75 |
| **Age** | | |
| 18-20 | 13 | 12 |
| 21-23 | 86 | 79.6 |
| 24-26 | 9 | 8.3 |
| **Program** | | |
| Bsc Information Technology | 55 | 50.9 |
| Bsc Computer Science | 2 | 1.9 |
| Bsc Multimedia | 47 | 43.5 |
| Bsc Education | 1 | 0.9 |
| Bsc Business Mathematics | 2 | 1.9 |
| Bsc Networking | 1 | 0.9 |
| **Course Subjects** | | |
| Database | 63 | 58.3 |
| Introduction to Programming Java | 28 | 25.9 |
| System Analysis and Design | 3 | 2.8 |
| Basic Programming | 5 | 4.6 |
| Expert System | 3 | 2.8 |
| Software Engineering | 2 | 1.9 |
| Artificial Intelligence | 2 | 1.9 |
| Basic Networking | 2 | 1.9 |
| **Semester** | | |
| Semester 1 | 5 | 4.6 |
| Semester 2 | 24 | 22.2 |
| Semester 3 | 20 | 18.5 |
| Semester 4 | 54 | 50.0 |
| Semester 6 | 5 | 4.6 |

From the 108 participants, 55 (50.9%) of them were undergoing Bsc. Information Technology program, while only 2 (1.2%) participants were studying Bsc. Computer Science. The other 47 (43.5%) participants enrolled for Bsc. Multimedia. The remaining are 1 (0.9%) for Bsc. Education, 2 (1.9%) for Bsc. Business Mathematics, and 1 (0.9%) for Bsc Networking. In detail, during the data collection, 63 (58.3%) participants took Database course, the largest percentage. It is followed by Introduction to Java Programming (28 (25.9%) participants), Basic Programming (5 (4.6%) participants), System Analysis and Design and Expert System (both 3 (2.8%) participants), and Software Engineering, Artificial Intelligence, and Basic Networking courses (all with 2 (1.9%) participants). The statistics also show that five (4.6%) participants were in semester 1, while 24 (22.2%) respondents were in semester 2. The other 20 (18.5%) respondents were in semester 3, 54 (50%) in semester 4 and 5 (4.6%) respondents were in semester 6.

On the other hand, the descriptive statistics of experiment 2 (YSYR) as shown in Table 4.2 reveals that eight (34.8%) participants were male while 15 (65.2%) participants were female. Most participants were between 21 and 23 years old (14 (60.9%) participants). Only six (26.1%) participants were between 18 and 20 years old, and three (13%) between 24 and 26 years old. Most of them enrolled for Bsc. Information Technology (18 (78.3%) participants) while the rest (five (21.7%) participants) enrolled for Bsc. Multimedia.

Table 4.2 also shows that 9 (39.1%) participants took Database course while 11 (47.8%) participants took Introduction to Java Programming. Meanwhile, only 1

(4.3%) participant took Programming Enhancement Program, Expert System, and Basic Networking courses. Besides that, 4 (17.4%) participants were in semester 2, 5 (21.7%) in semester 3, 10 (43.5%) in semester 4, 3 (13%) in semester 6, and 1 (4.3%) in semester 9.

Table 4.2

*Demographic Statistics of Experiment 2 (YSYR)*

| **Variables / Factors** | **Frequency** | **Percentage** |
|---|---|---|
| **Gender** | | |
| Male | 8 | 34.8 |
| Female | 15 | 65.2 |
| **Age** | | |
| 18-20 | 6 | 26.1 |
| 21-23 | 14 | 60.9 |
| 24-26 | 3 | 13.0 |
| **Program** | | |
| Bsc Information Technology | 18 | 78.3 |
| Bsc Multimedia | 5 | 21.7 |
| **Course Subjects** | | |
| Database | 9 | 39.1 |
| Introduction to Programming Java | 11 | 47.8 |
| Programming enhancement program | 1 | 4.3 |
| Expert System | 1 | 4.3 |
| Basic Networking | 1 | 4.3 |
| **Semester** | | |
| Semester 2 | 4 | 17.4 |
| Semester 3 | 5 | 21.7 |
| Semester 4 | 10 | 43.5 |
| Semester 6 | 3 | 13.0 |
| Semester 9 | 1 | 4.3 |

## 4.3 Structural Equation Modeling

Structural Equation Modeling (SEM) is a methodological technique to ease the analytical complex model. Thus, SEM is a statistical technique for addressing a confirmatory approach of a structural theory that generates observation on multiple variables (Bentler, 1988; Barbara, 2010). Researches have shown that there are two types of SEM named as the Covariance-Based SEM (CB-SEM) and Partial Least Square SEM (PLS-SEM). The CB-SEM is purposely for estimating the parameters of the model in order to reduce the variation between the sample covariance and those predicted by the theoretical model. It reduces the effort to predict the existence of dependent variables through the maximisation of the variance explained ($R^2$) of the dependent variable (Barroso et al., 2010). In contrast, PLS-SEM is capable of making use of both normal and non-normal dataset. Hence, this study uses PLS-SEM to analyze the collected data.

## 4.4 Analytical Activities in Structural Equation Modeling

The assessment of PLS-SEM covers two different approaches specifically for achieving different objectives, which are measurement model and structural model assessments (Wilson, 2010). The first approach is known as the measurement model evaluation, which addresses the reliability and validity of measures that form embedded constructs (Wilson, 2010; Chin, 2010). In detail, Hair et al. (2010) and Chin (2010) emphasizes that major activities in evaluating the measurement model are internal consistency reliability, indicator reliability, convergent validity, and discriminant validity (Lewis, Templeton, & Byrd, 2005; Straub, Boudreau, & Gefen, 2004). Besides that, structural model analytical phase in SEM also addresses the significance of the path coefficients and level of $R^2$ (Hair et al., 2010; Chin, 2010).

### 4.4.1 Reliability of Internal Consistency

Within PLS, composite reliability (CR) is used to measure the **internal consistency** (Chin, 1998). CR takes into consideration the difference in loadings of the indicators (Hashim, 2012). The reliability of an internal consistency is deemed satisfactory when the value is at the minimum level (0.7) in the early stage of research and increase to 0.8 or 0.9 in the later stages. Any value below 0.6 reflects a lack of reliability (Nunnally, 1994). For this study, the CR for each construct is shown in Tables 4.3 and 4.4, which are greater than 0.7. This indicates that the internal consistency is satisfactory.

Table 4.3

*Descriptive and Reliability Statistics for NSYR*

| Construct | Item | Mean | Standard Deviation | Loading | T-Statistics |
|---|---|---|---|---|---|
| Socialization CR= 0.8697 | SF1 | 4.00 | .820 | 0.9455 | 2.8634 |
| | SF2 | 4.18 | .818 | 0.7164 | 2.4294 |
| | SF3 | 4.23 | .793 | 0.5371 | 1.5798 |
| | SF4 | 4.32 | .734 | 0.7149 | 2.4504 |
| | SF5 | 4.12 | .872 | 0.8312 | 2.9089 |
| Externalization CR= 0.741 | E1 | 3.63 | 1.010 | 0.9039 | 2.0797 |
| | E2 | 4.09 | .803 | 0.5605 | 1.3762 |
| | E4 | 3.87 | .928 | 0.6071 | 1.4378 |
| Combination CR= 0.7912 | C2 | 4.08 | .866 | 0.4411 | 1.2179 |
| | C4 | 3.40 | 1.160 | 0.8466 | 2.6186 |
| | C5 | 3.57 | 1.070 | 0.9053 | 2.6838 |
| Internalization CR= 0.8767 | IIODM1 | 3.65 | .889 | 0.888 | 3.1684 |
| | IIODM2 | 3.09 | .981 | 0.6358 | 1.9121 |
| | IIODM3 | 3.74 | .741 | 0.8341 | 3.1045 |
| | IIODM 5 | 3.58 | .844 | 0.7677 | 2.979 |
| | IIOT3 | 3.95 | .847 | 0.5645 | 1.6693 |
| | IIOT4 | 3.94 | .795 | 0.6575 | 2.2835 |
| | IIOT5 | 3.93 | .817 | 0.5877 | 1.831 |
| Code Quality (NSYR) | NSYR | 4.50 | 1.204 | 1 | 0 |

Table 4.4

*Descriptive and Reliability Statistics for YSYR*

| **Construct** | **Item** | **Mean** | **Standard Deviation** | **Loading** | **T-Statistics** |
|---|---|---|---|---|---|
| Socialization CR= 0.9186 | SF1 | 4.13 | .920 | 0.5775 | 1.8629 |
| | SF2 | 4.30 | .703 | 0.9478 | 3.9258 |
| | SF3 | 4.35 | .714 | 0.9201 | 3.3777 |
| | SF4 | 4.30 | .703 | 0.8562 | 3.0366 |
| | SF5 | 3.91 | 1.083 | 0.8243 | 3.7991 |
| Externalization CR= 0.7682 | E2 | 4.17 | .778 | 0.5145 | 1.207 |
| | E4 | 3.70 | 1.105 | 0.8784 | 2.297 |
| Combination CR= 0.7805 | C4 | 3.52 | 1.238 | 0.1697 | 0.3843 |
| | C5 | 3.43 | 1.199 | 0.9952 | 4.5707 |
| Internalization CR= 0.923 | IIODM1 | 3.78 | .902 | 0.8559 | 3.2322 |
| | IIODM2 | 4.09 | .733 | 0.6017 | 2.1824 |
| | IIODM3 | 3.43 | 1.161 | 0.7347 | 2.6849 |
| | IIODM 5 | 4.00 | .739 | 0.6149 | 1.982 |
| | IIOT3 | 4.04 | .767 | 0.7181 | 2.1625 |
| | IIOT4 | 3.91 | .900 | 0.753 | 2.202 |
| | IIOT5 | 3.96 | .767 | 0.7438 | 2.6714 |
| | IIOL2 | 3.17 | .885 | 0.6689 | 2.2139 |
| | IIOL3 | 4.04 | 1.054 | 0.6232 | 2.2286 |
| | IIOL5 | 3.74 | 1.114 | 0.8432 | 3.1003 |
| | IIOL7 | 3.65 | .878 | 0.7581 | 2.4053 |
| Code Quality (YSYR) | YSYR | 3.57 | 2.233 | 1 | 0 |

## 4.4.2 Indicator Reliability

In order to assess **indicators' reliability**, the researcher needs to evaluate to what extent a variable or a group of variables is proportionate with what it means to measure (Urbach & Ahlemann, 2010). The reliability construct is evaluated independently from other constructs. With reference to Chin (1998), **indicator loadings** must be significant at minimum 0.05 and the **loading** should be greater than 0.7. This is because with the loading value at 0.7, a latent variable (LV) is considered to be able to explain at least 50 percent of its indicator's variance. On the other hand, Bootstrapping is resampling method that can be used to examine the significance of the indicator loadings. In general, the decision of eliminating an indicator should be taken carefully when considering PLS characteristics of consistency (Henseler, Ringle, & Sinkovics, 2009). In case of low value of an indicator, it is logic to take the decision of eliminating that indicator and that elimination is linked with the significant increase of CR value (Hashim, 2012). Therefore, the indicator reliability in NSYR model ranges from 0.741 to 0.8767 as shown in Table 4.3 and in YSYR model, the indicator reliability is ranges from 0.7682 to 0.923as shown in Table 4.4.

## 4.4.3 Convergent validity

Convergent validity indicates the extent to which individual items reflect a construct converging as compared with items that measure various constructs (Urbach & Ahlemann, 2010). With the aid of PLS, the value of average variance extracted (AVE) is used to calculate the convergent validity. According to Fornell and Larcker

(1981), in case of AVE value of a construct amount is at least 0.5, then the convergent validity is considered sufficient.

In regards to that, the convergent reliability for NSYS model for this study is exhibited in Table 4.5. It reveals that the entire construct AVE value are above the threshold value (0.5). In the context of this research, the AVE ranges from 0.5 to 0.5796. This shows that the analysis satisfies the AVE rule.

Table 4.5

*AVE Values of NSYS Model*

| **Construct** | **AVE** |
|---|---|
| Code Quality (NSYR) | 1 |
| Combination | 0.577 |
| Externalization | 0.5 |
| Internalization | 0.5106 |
| Socialization | 0.5796 |

Further, the CR for YSYR model is shown in Table 4.6 and reveals that the entire construct AVE value are above the threshold value (0.5). In the context of this research, the AVE is ranges from 0.5097 to 0.6982. This also satisfies the AVE rule.

Table 4.6

*AVE Values of YSYS Model*

| **Construct** | **AVE** |
|---|---|
| Code Quality (YSYR) | 1 |
| Combination | 0.5097 |
| Externalization | 0.5182 |
| Internalization | 0.5245 |
| Socialization | 0.6982 |

### 4.4.4 Discriminant validity

Discriminant validity is used to distinguish one measure from another of a construct measures. On the contrary, the convergent validity, discriminant validity examines whether the items intentionally measure another issue (Urbach & Ahlemann, 2010). Within PLS, cross loading (Chin, 1998) and standard of Fornell-Larcker (Fornell & Larcker, 1981) are two commonly used measures of discriminant validity. The first measurement analysis was conducted by examining the AVE for both YSYR and NSYR models and represented in Tables 4.7 and 4.8.

Table 4.7

*Discriminant Validity for YSYR*

| | Code Quality (YSYR) | Combination | Externalization | Internalization | Socialization |
|---|---|---|---|---|---|
| Code Quality (YSYR) | **1** | 0 | 0 | 0 | 0 |
| Combination | -0.4662 | **0.7139** | 0 | 0 | 0 |
| Externalization | -0.2896 | 0.5192 | **0.7199** | 0 | 0 |
| Internalization | 0.2918 | 0.414 | 0.3571 | **0.7242** | 0 |
| Socialization | -0.2548 | 0.3401 | 0.4254 | 0.6092 | **0.8356** |

Table 4.8

*Discriminant Validity for NSYR*

| | Code Quality (NSYR) | Combination | Externalization | Internalization | Socialization |
|---|---|---|---|---|---|
| Code Quality (NSYR) | **1** | 0 | 0 | 0 | 0 |
| Combination | -0.2217 | **0.7596** | 0 | 0 | 0 |
| Externalization | 0.0904 | 0.3718 | **0.7071** | 0 | 0 |
| Internalization | 0.2378 | 0.1966 | 0.4487 | **0.7146** | 0 |
| Socialization | 0.1049 | 0.3022 | 0.6308 | 0.4983 | **0.7613** |

The next form of assessing the discriminant validity is through the cross loading of the indicators where by the exact items are larger than its cross loading. This is achieved in this study through the results in Tables 4.9 and 4.10.

Table 4.9

*Discriminant Validity (Cross Loading Criterion) For NSYR Model*

| | Code Quality (NSYR) | Combination | Externalization | Internalization | Socialization |
|---|---|---|---|---|---|
| C2 | -0.0061 | **0.4411** | 0.5181 | 0.4217 | 0.5487 |
| C4 | -0.1715 | **0.8466** | 0.3834 | 0.2148 | 0.2468 |
| C5 | -0.2162 | **0.9053** | 0.2723 | 0.1301 | 0.2691 |
| E1 | 0.0939 | 0.2827 | **0.9039** | 0.3628 | 0.5237 |
| E2 | 0.0312 | 0.2023 | **0.5605** | 0.4222 | 0.5401 |
| E4 | 0.0421 | 0.3495 | **0.6071** | 0.2411 | 0.3484 |
| IIODM1 | 0.2527 | 0.0422 | 0.3909 | **0.888** | 0.478 |
| IIODM2 | 0.127 | 0.2285 | 0.0907 | **0.6358** | 0.1137 |
| IIODM3 | 0.1872 | 0.1877 | 0.4522 | **0.8341** | 0.384 |
| IIODM5 | 0.1424 | 0.2511 | 0.4242 | **0.7677** | 0.5156 |
| IIOT3 | -0.028 | 0.3624 | 0.5402 | **0.5645** | 0.5297 |
| IIOT4 | 0.0126 | 0.3547 | 0.5937 | **0.6575** | 0.6145 |
| IIOT5 | 0.0195 | 0.4128 | 0.5164 | **0.5877** | 0.6599 |
| NSYR | **1** | -0.2217 | 0.0904 | 0.2378 | 0.1049 |
| SF1 | 0.1257 | 0.218 | 0.5824 | 0.4789 | **0.9455** |
| SF2 | 0.0343 | 0.26 | 0.5176 | 0.4644 | **0.7164** |
| SF3 | 0 | 0.3272 | 0.5539 | 0.2279 | **0.5371** |
| SF4 | 0.0064 | 0.2846 | 0.4838 | 0.3856 | **0.7149** |
| SF5 | 0.0638 | 0.3546 | 0.5136 | 0.3382 | **0.8312** |

Table 4.10

*Discriminant validity (cross loading criterion) for YSYR model*

| | Code Quality (YSYR) | Combination | Externalization | Internalization | Socialization |
|---|---|---|---|---|---|
| C4 | -0.0462 | **0.1697** | 0.6739 | 0.2477 | 0.3644 |
| C5 | -0.4672 | **0.9952** | 0.4588 | 0.3944 | 0.3082 |
| E2 | -0.1435 | 0.8444 | **0.5145** | 0.6512 | 0.4236 |
| E4 | -0.2575 | 0.1343 | **0.8784** | 0.0532 | 0.2596 |
| IIODM1 | 0.2152 | 0.4358 | 0.381 | **0.8559** | 0.4252 |
| IIODM2 | 0.0453 | 0.3885 | 0.7413 | **0.6017** | 0.291 |
| IIODM3 | 0 | 0.8035 | 0.4507 | **0.7347** | 0.3912 |
| IIODM5 | -0.196 | 0.8686 | 0.4988 | **0.6149** | 0.5145 |
| IIOL2 | 0.0906 | 0.5472 | 0.1138 | **0.6689** | 0.4446 |
| IIOL3 | 0.0579 | 0.7501 | 0.4458 | **0.6232** | 0.3416 |
| IIOL5 | 0.1044 | 0.6011 | 0.5082 | **0.8432** | 0.5472 |
| IIOL7 | 0.1845 | 0.4253 | 0.2301 | **0.7581** | 0.6157 |
| IIOT3 | 0 | 0.3539 | 0.4753 | **0.7181** | 0.7049 |
| IIOT4 | 0.1291 | 0.2709 | 0.2658 | **0.753** | 0.4665 |
| IIOT5 | 0.0707 | 0.3351 | 0.2566 | **0.7438** | 0.7005 |
| SF1 | 0 | 0.233 | 0.3496 | 0.6761 | **0.5775** |
| SF2 | -0.333 | 0.3381 | 0.3664 | 0.5029 | **0.9478** |
| SF3 | -0.1641 | 0.1814 | 0.3175 | 0.5246 | **0.9201** |
| SF4 | -0.0833 | 0.1135 | 0.1786 | 0.5947 | **0.8562** |
| SF5 | -0.163 | 0.4714 | 0.595 | 0.6531 | **0.8243** |
| YSYR | **1** | -0.4662 | -0.2896 | 0.2918 | -0.2548 |

## 4.5 Validation of Structural Model

Validation of the structural model can assist the researcher to systematically estimate whether the data support the hypotheses characterized by the structural model (Urbach & Ahlemann, 2010). It is not proper to establish the analysis of the structural model unless the measurement model has been achieved successfully.

Within PLS, a coefficient of determination ($R^2$), and path coefficients are used to evaluate the structural model.

### 4.5.1 Coefficient of Determination ($R^2$)

The variance explanation of $R^2$ measures the relationship of latent variables to its total variance. Based on the benchmark by Chin (1998), $R^2$ is considered weak if it is 0.19 and below. $R^2$ of 0.333 is accepted as the average, while $R^2$ of 0.67 is considered as substantial. The Figures 4.1 and 4.2 represent the results of structural model for NSYR and YSYR obtained in this study respectively.

With reference to Figure 4.1, Socialization, Externalization, Combination, and Internalization are able to explain 13.7% of the variance in code quality of NSYR. This shows that coefficient of determination $R^2$ is weak. On the hand, Socialization, Externalization, Combination, and Internalization are able to explain 72.4% of the variance on code quality of YSYR .

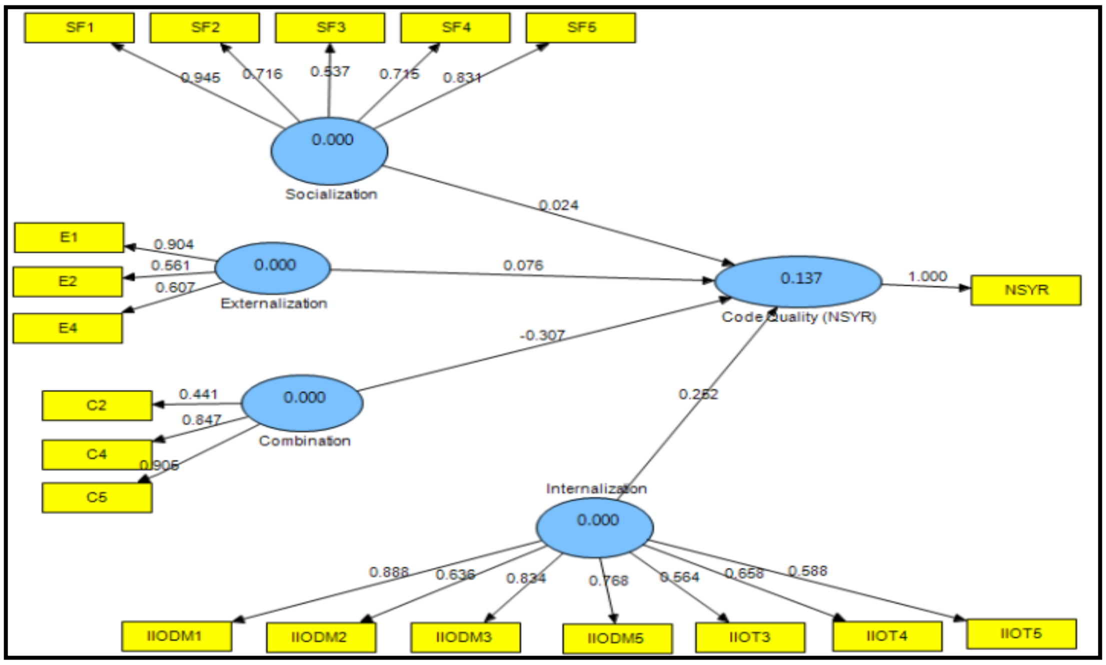


*Figure 4.1.* Result of NSYR structural model

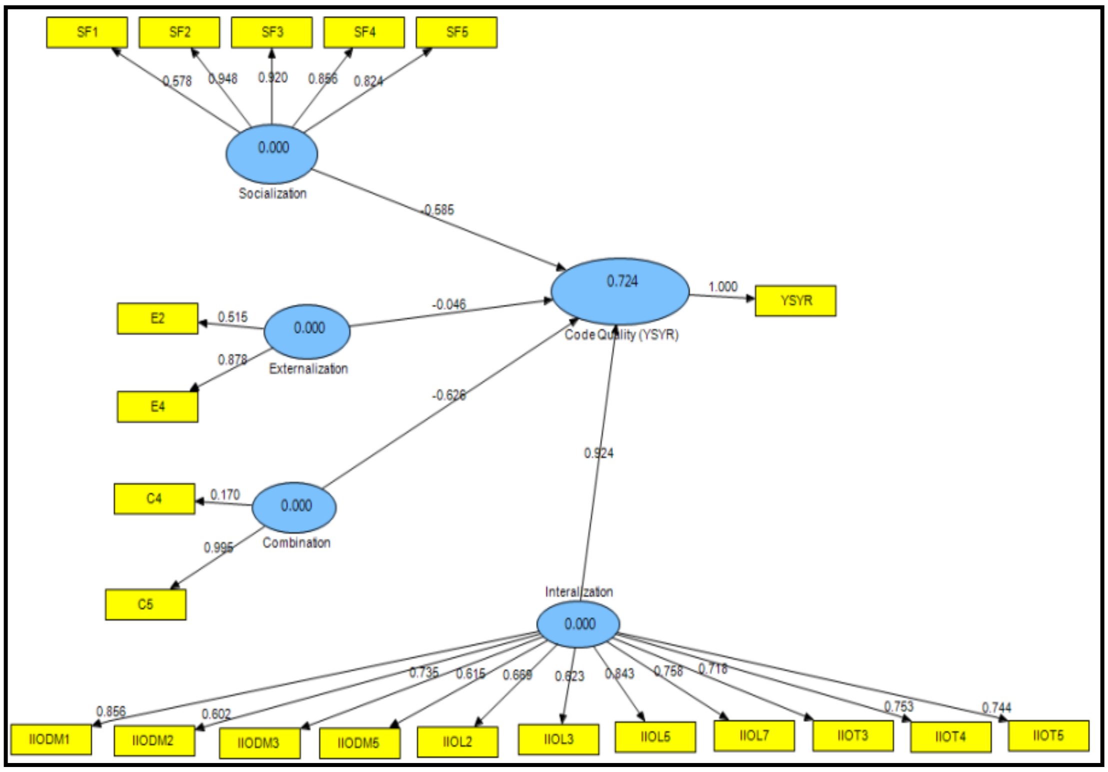


*Figure 4.2. Result of YSYR structural model*

### 4.5.2 Path Coefficient

By testing the path coefficient value, a researcher is eligible to know whether the relationship between two LV is strong enough. In order to investigate the relationship between two LVs, the researcher needs to notice the path coefficients, algebraic sign, magnitude, and significance. According to Huber et al. (2007), the impact of the model would be felt if the path coefficient is greater than 0.100 and significant to support the hypothesis at 0.05 significant level. For this study, the rules for validating the structural model are illustrated in Table 4.13.

The standard values for assessing the measurement model of this study are shown in Table 4.11. In which, T-test values helped the researcher to judge which of the hypothesis are  supported. According to Chin (2008), when T-test is above or equal 0.9 thus the hypothesis is supported, otherwise the hypothesis is not supported when T-test value is less than 0.9.

Table 4.11

*Standard Values for Assessing Measurement Model*

| Dependent Variable | Independent Variable | Path Coefficient ( ) | Observed T- statistics | Significance Level |
|---|---|---|---|---|
| Code Quality (YSYR) | Socialization | -0.5852 | 3.5097 | 0.05 |
| | Externalization | -0.0456 | 0.3025 | 0.05 |
| | Combination | -0.6258 | 2.0617 | 0.05 |
| | Internalization | 0.9236 | 2.4107 | 0.05 |
| Code Quality (NSYR) | Socialization | 0.0241 | 2.0776 | 0.05 |
| | Externalization | 0.0762 | 0.2164 | 0.05 |
| | Combination | -0.3069 | 2.8001 | 0.05 |
| | Internalization | 0.2519 | 1.6609 | 0.05 |

Table 4.12 shows the supported hypotheses for this study based on the outcomes given by Table 4.11

Table 4.12

*Supported Hypotheses of The Study*

| Hypothesis | Codification | Description | Result |
|---|---|---|---|
| H1 | $S \xrightarrow{NS} CQ$ | The Socialization process contributes positively to code quality without employing SECI process. | Supported |
| H2 | $E \xrightarrow{NS} CQ$ | The Externalization process contributes positively to code quality without employing SECI process. | Not Supported |
| H3 | $C \xrightarrow{NS} CQ$ | The Combination process contributes positively to code quality without employing SECI process. | Supported |
| H4 | $I \xrightarrow{NS} CQ$ | The Internalization process contributes positively to code quality without employing SECI process. | Supported |
| H5 | $S \xrightarrow{YS} CQ$ | The Socialization process contributes positively to code quality with employing SECI process. | Supported |
| H6 | $E \xrightarrow{YS} CQ$ | The Externalization process contributes positively to code quality with employing SECI process. | Not Supported |
| H7 | $C \xrightarrow{YS} CQ$ | The Combination process contributes positively to code quality with employing SECI process. | Supported |
| H8 | $I \xrightarrow{YS} CQ$ | The Internalization process contributes positively to code quality with employing SECI process. | Supported |

Table 4.13

*Standard Values for Validating Structural Model*

| **Assessment Subjects** | **Measures** | **Threshold Values** |
|---|---|---|
| Internal Consistency Reliability | Composite Reliability | > 0.7 |
| Indicator Reliability | Factor Loadings | > 0.7 |
| Convergent Validity | Average Variance Extracted (AVE) | > 0.5 |
| Discriminant Validity | Fornell-Larcker Criterion | - |
| Discriminant Validity | Loading – Cross-loadings Comparison | - |

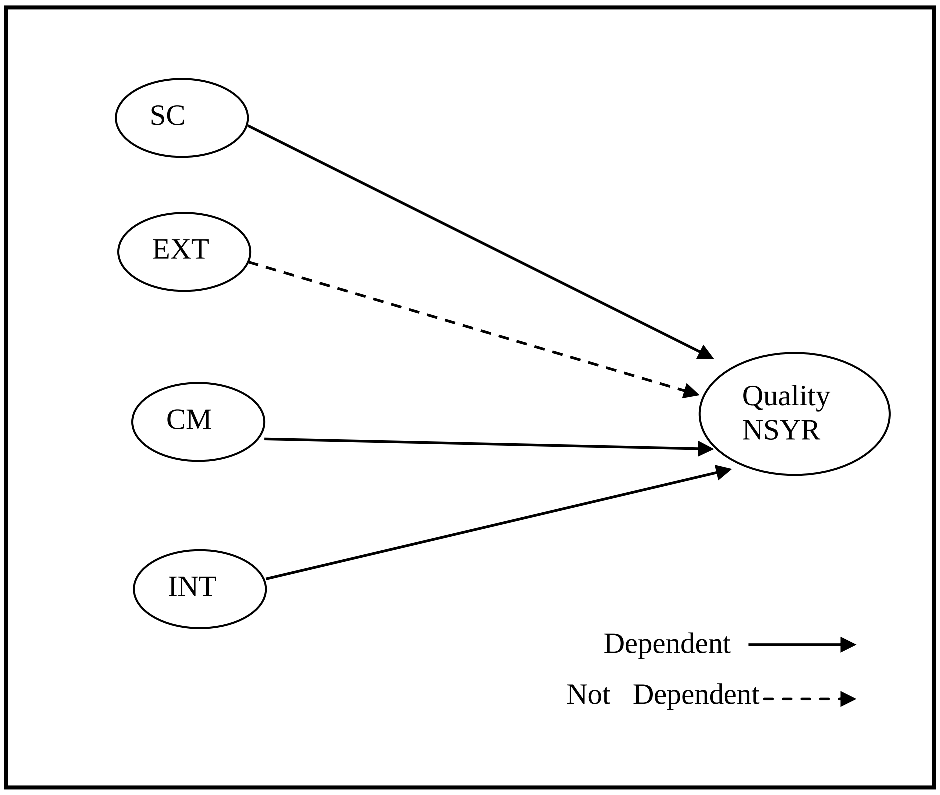


*Figure 4.3.* Results of the hypothesis for experiment 1 (NSYR)

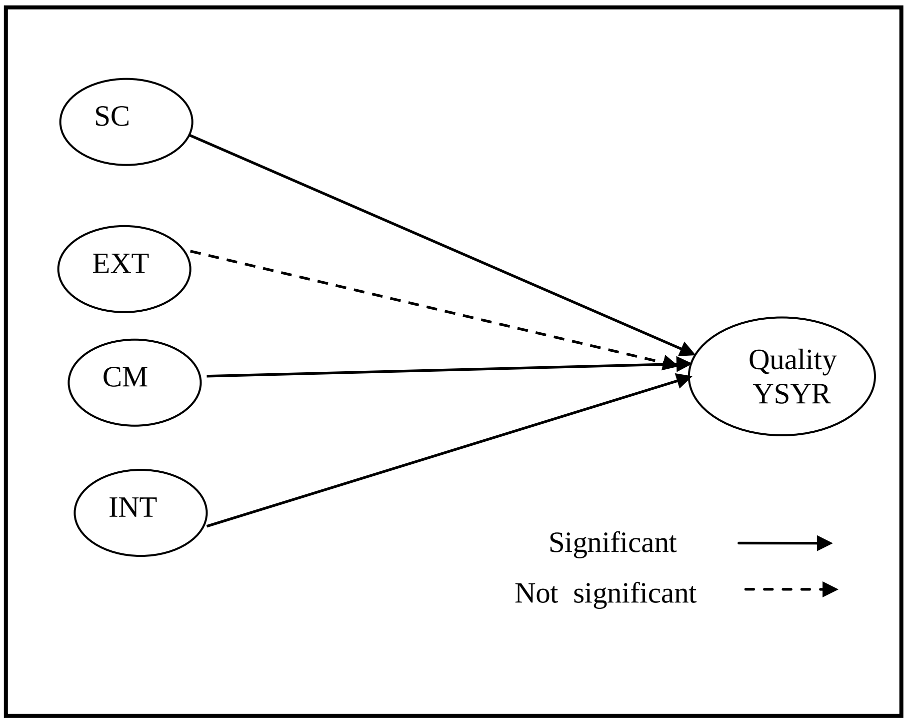


*Figure 4.4.* Results of the hypothesis for experiment 2 (YSYR)

## 4.6 Research Model

The study has been carried out as outlined in the research model illustrated in Figure 4.5.

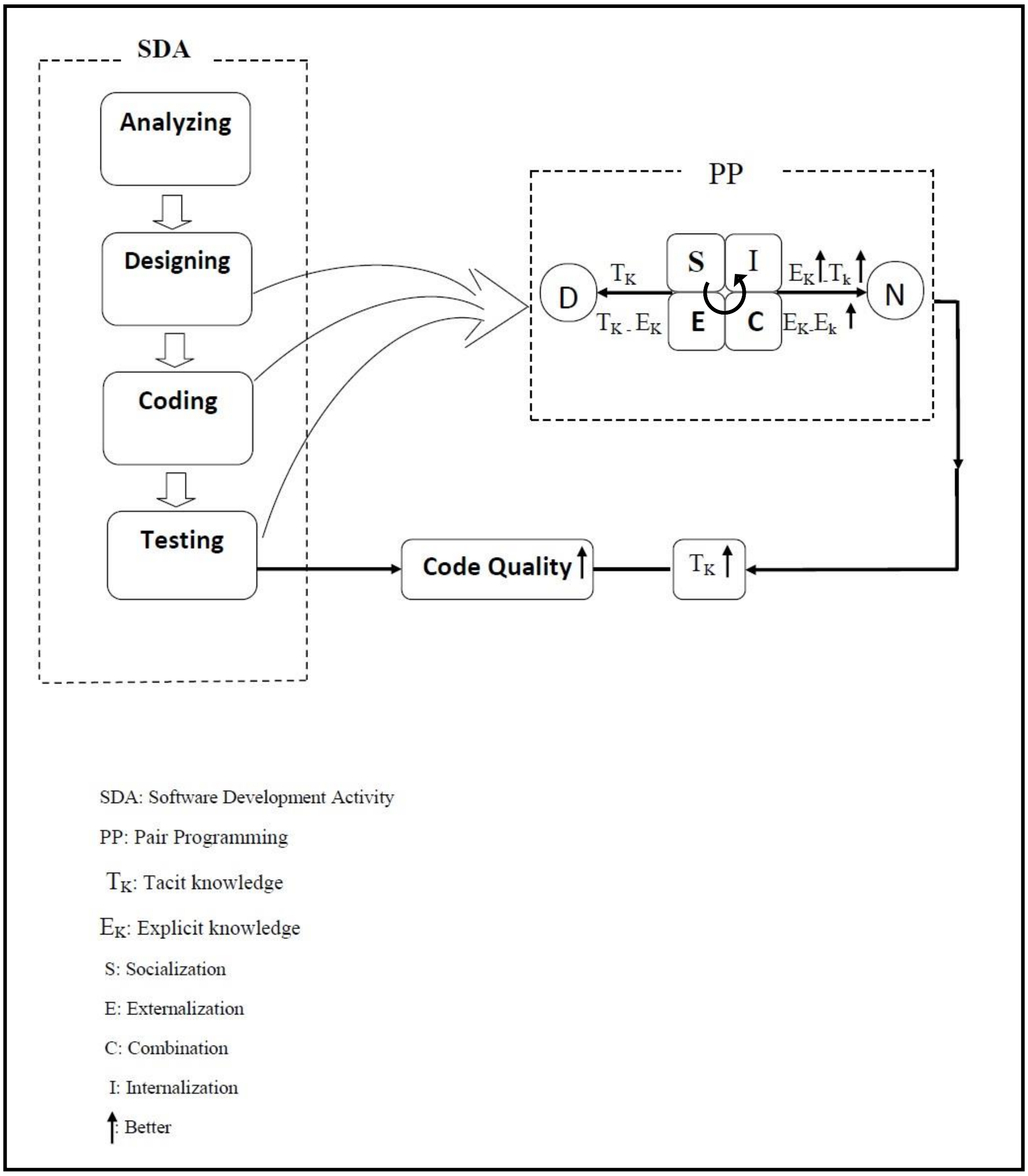


*Figure 4.5.* Research model

The research model in Figure 4.5 illustrates four common stages of software development life cycle. Particularly, PP practices are commonly applied in designing, coding, and testing phases. In the practice, the driver and the navigator interact simultaneously to manage same issue. In this study, the conceptual model investigates the interaction between the driver and the navigator and its impact on tacit knowledge transferred from one student to another based on SECI model. Above of that, it tracks the performance of students in term of quality of end-program, based on marks given by the instructor.

## 4.7 Summary

This chapter discusses and analyzes the results of experiments. The procedures are explained in detail, including the instrumentation. Besides, the survey is also explained. The results are discussed in detail too, explaining that the hypotheses are supported unless hypotheses two and six. A descriptive statistics of the respondent is also conveyed. Indeed, the measurement model through the evaluation of the reliability and validity of measures is addressed. Finally, the evaluation of the structural model is explained leading to the results of the hypothesis and variance explanation of the research model.

# CHAPTER FIVE
# DISCUSSION AND CONCLUSION

## 5.1 Introduction

This chapter discusses the findings base on the results obtained in Chapter 4. Firstly, an overview of the research is presented. Then, discussions on the hypotheses testing for the hypotheses based on the output generated from SEM of both experiment 1 and 2 are presented. Next, the revised model for successful PP knowledge-based, which is based on the significant or supported results obtained from the hypotheses is presented. Further, the chapter summarizes and reviews the findings that were obtained. Besides that, the research presents contribution to achieve viable goals based on the described problem and objectives (in Chapter 1). Finally, the chapter discusses the limitations and the direction of future works.

## 5.2 Overview of the Research

The research is conducted mainly to propose a conceptual model of PP knowledge-based sharing for improving programming skills. Assessment of the factors for achieving a viable PP knowledge-based model has become necessary due to the revelations of previous studies that Agile development does not allow a successful knowledge sharing environment for team members. Thus, it is mandatory to identify the influencing factors for achieving an effective PP knowledge-based model which allows team work in the knowledge sharing environments. In view of that, an effective PP knowledge-based model is proposed and validated in order to identify the influential factors.

Based on the literatures, it has been found that various studies have suggested that the potential influencing factors for achieving PP knowledge-based sharing model for improving programming skills are socialization (SC), Externalization (EXT), Combination (CMB), Independent of learning (INDL), Independent of Thinking (INDT) and Independent of Decision Making (INDDM). Based on these factors, this study develops an effective PP knowledge-based model for improving programming skills and shows the relationships among the significant drivers or factors. Finally, a modified holistic model of learning is proposed in the context of higher learning institution.

Path analysis with SEM using SmartPLS 2.0 software was used to test all the hypothesized relationships in the structural model. The results of the proposed hypotheses are shown in Tables 4.14 and 4.15 respectively. Besides, the findings of the hypothesized results are discussed, while the findings from the previous studies are used to support or disprove the significance of the findings of this study. Hence, the subsequent sections present the discussions on the outcome of hypotheses testing.

## 5.3 Discussion of the results based on the objectives:

The experiments were divided into two groups with the dependent variable of code quality of the first experiment named NSYR and the dependent variable of code quality of the second experiment tagged as YSYR. Meanwhile, the independent variables are uniform for both groups as socialization (SC), Externalization (EXT), Combination (CMB), Internalization (INT). The following sub-sections discuss the results on the basis of the three objectives of this study

### 5.3.1 Discussion based on the first objective:

This suction discuss the results based on the first objective of the study that concerns in investigating the relationships between each of the four processes of SECI model and code quality. The effect of each stage on the code quality is outlined in the following sub-sections.

#### a. Effect of Socialization on Code Quality

It is generally believed that interaction or sharing of knowledge in a virtual way or from tacit to tacit form may not yield full understanding to the listeners or pair group based on the individual intelligence level. The literatures reveal that there is a relationship between the sharing of knowledge in the form of tacit to tacit between two people or groups towards achieving a code quality. This is confirmed by the results of the two experiments with model and without model (YSYR and NSYR). The implication of this result in the student without model (NSYR) is that participants have found to have prior or basic knowledge of Java programming language.

This enables them to transfer the knowledge between PP members without documentation and achieve code quality. In the context of PP laboratory assignment (YSYR), the results show that it would be easier for the participants to achieve code quality. This is as a result of their exposure to the knowledge of JAVA programming language. These explain that socialization is significantly related to code quality and in line with the study by Singh et al. (2013). Comparatively, the relationship between the Socialization process and code quality by the participants with model (YSYR) is better than the relationship between Socialization and code quality by the participant without model (NSYR) (t values = 3.5097 and 2.0776 respectively).

**b. Effect of Externalization on Code Quality**

Based on the obtained result from the analysis of the collected data in the two experiments (NSYR and YSYR), the results reveal that there is not significant relationship between the driver and the navigator. The obtained result is consistent with the study by Ahmad et al. (2012), which affirmed that achieving a project's completion is transfer of knowledge from abstract to documented form does not bring about the code quality. Additionally, the result of the hypothesis may trace to lack of participant to write draft code before simply writing the codes in the computer. On the other hand, the Externalization towards code quality in NSYR and YSYR ended not significant. However, the result of YSYR is better than NSYR with t value = 0.3025 and 0.2164 respectively.

### c. Effect of Combination on Code Quality

Combination is one of the knowledge management models, which focuses on sharing or transferring of knowledge between the pair from explicit format to explicit format. The obtained results in both experiments 1 and 2 (YSYR and NSYR) as shown in Table 4.12 support the statement that the relationship between Combination and code quality is significant. This means that it is mandatory to document the references that guide the code quality could be achieved through the Combination form knowledge transfer. Hence, the obtained result is consistent with the previous study by Ahmad et al. (2012). The implication of this finding is it provides people who are involved in the learning and sharing of programming skill to develop quality code, should they have access to references while writing the codes for the given assignment. Besides that, the comparison between the two programming assignments shows that assignment without model is better than assignment with model. This is deduced based on the t value = 2.8001 and 2.0617 respectively.

### d. Effect of Internalization on Code Quality

Internalization in the SECI model is described as systematic explicit knowledge, which can be converted into a richer consistent and more complicated tacit knowledge, such as saved in human memory (memorization). It was initially hypothesized that there is a significant relationship between knowledge shared from concrete to an abstract form when determining or seeking for programming skills. In the context of this research, both analyses in the two experiments (with model and without model) confirm that there are significant relationships between Internalization and code quality of Java programming assignment. The obtained findings are in line with the previous studies that support the hypothesized statement (Cayaba & Pablo, 2013; Jiangping et al., 2013; Mawarny, Mazni, Mazida, & Khairul, 2011). This implies that exchange of knowledge from explicit form to tacit form while addressing Java programming language helps in achieving code quality. Finally, the comparison of results of the two experiments show that YSYR is better than NSYR through t values = 2.4107 and 1.6609 respectively.

Conclusively, the significant findings among the four research hypotheses show that only one construct is agreeably not supported in the two experiments, which is Externalization. This implies that Socialization, Combination and Internalization are the determinant factors of code quality while modeling PP knowledge-base and represented Figure 5.1. In addition, Socialization in YSYR is found as the most influential factor among the SECI processes.

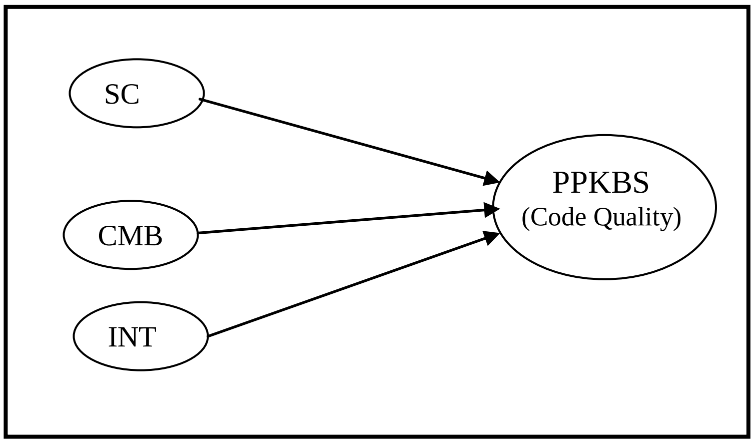


*Figure 5.1.* A revised model for PP knowledge-based sharing for improving programming skills

### 5.3.2 Discussion based on the second objective:

The proposed model is constructed on the basis of achievement of the first objective which is to investigate the relationships between code quality and each of the four factors of knowledge management which are socialization, externalization, combination and internalization. In which, the interaction between pair programmers, who are driver and navigator, to find a programing solution relies on the factors of knowledge management to transfer knowledge between the pair programmers. The positive interaction between pair programmers leads to improve their programming skills and thus achieving better code quality based on the outcomes from the first objective.

### 5.3.3 Discussion based on the third objective:

The third objective which is validate the proposed model was achieved in this study through employing Sequential Equation Modeling SEM for the analysis and interpretation phase  to test the hypothesis and examining whether the conceptual model fits the collected data through the experiment. In addition, this study utilizes smart partial least square PLS2.0 tool to avert the limitations of co-variance besides SEM. Above of that, PLS is suitable for small sampling size which is commonly less than 30. Smart PLS was used because it produces the statistical assessment for the measurement model that enabled the researcher to decide which are the influential factors  and which of the factors is the most contributor. In this study, there are three influential factors for the model of pair programming knowledge based sharing for improving programming skills which are Socialization, Combination and

Internalization. Socialization between the driver and navigator is the most contributor in improving the code quality.

## 5.4 Study Contribution

This study has contributed in providing a road map for the educators to achieve code quality using effective teaching methods through determining the impact factors for determining PP knowledge-based sharing for improving programming skills.. Above of that, this study provides the empirical evidence on the impact of each Socialization, Combination and Internalization on code quality

When the educators know that Socialization can provide a significant impact on code quality, they will be able to emphasize more discussion among pairs in the teaching method.

The positive impact of Combination on code quality leads the educators to the importance of providing various sources while teaching process meaning that students are allowed to rely on different teaching materials other than the notes given in the lecture such as internet, books and any helpful notes. In addition, this study highlights the positive contribution of internalization in providing better code quality that give the educators to tackle for enriching the knowledge of the students.

This study comes up with a reliable research instrument to be used for the future research in the domain of technology innovative at higher learning institution as shown in Appendix A.

## 5.5 Problems and Limitation

Despite the fact that this study provides the stakeholders the main factors they need to focus while searching effective code quality, the study still faces the following set-back:

- The study is able to gather only 23 participants from the second experiment due to the time constraints and the participation of the students under voluntary basis. Majority of the students had mid-term exams while second experiment was held.
- The application of the results of this research is limited to the University Utara Malaysia, while more cases of data are needed to be collected for generalization of the research findings.

## 5.6 Future Work

As it was stated that this research provides the stakeholders at higher learning institution, the needs to achieve effective program code quality. Consequently, the following improvement needs to be addressed in future research:

- The number of participants needs to be increased in the future research in order to achieve robust results.
- The qualitative research approach should be added to the work in order to obtain full representative of the participants' mind.

## 5.7 Recommendation

The importance of achieving a code code quality while dealing with PP knowledge sharing at higher learning institution cannot be overemphasized. Therefore, this calls for immediate recommendation of this research at higher learning institution since the research has identified the influential factors for achieving program code quality and knowledge sharing PP.

## 5.8 Summary

This chapter presents a lengthy discussion about the findings of hypothesized relationship between independent and dependent variables. A total of 9 hypotheses have been tested in achieving the objectives of the research. From the 4 tested hypotheses, only 1 hypothesis (Externalization) was agreeably not supported by the two experiments which lead to the formation of the revised model as shown in Figure 5.1. Besides, this chapter states the contribution of this study in achieving a program code quality and knowledge sharing PP.  On top of that, it also discusses the limitation of the study, expected future work, and recommendations.

# REFERENCES

Abidi, S. S. R., Cheah, Y.-N., & Curran, J. (2005). A knowledge creation info-structure to acquire and crystallize the tacit knowledge of health-care experts. *Information Technology in Biomedicine, IEEE Transactions on, 9*(2), 193-204.

Abran, A., Moore, J. W., Bourque, P., Dupuis, R., & Tripp, L. (2004). Guide to the software engineering body of knowledge, 2004 version. *IEEE Computer Society, 1*.

Aggestam, L. (2006). Learning organization or knowledge management–which came first, the chicken or the egg. *Information technology and control, 35*(3A), 295-302.

Ahmad, M., Md Rejab, M., Syazwan Abdulah, M., Omar, M., Bariah Ahmad, K., & Abbas, M. (2012). Measuring tacit knowledge acquired during problem based learning teaching method in learning management system environment. *AWERProcedia Information Technology and Computer Science, 1*.

Alegre, J., Sengupta, K., & Lapiedra, R. (2013). Knowledge management and innovation performance in a high-tech SMEs industry. *International Small Business Journal, 31*(4), 454-470.

Alipour, F., Idris, K., & Karimi, R. (2011). Knowledge Creation and Transfer: Role of Learning Organization. *International Journal of Business Administration, 2*(3).

Ally, M. (2004). Designing effective learning objects. *Online education using learning objects*, 87-97.

Anantatmula, V., & Kanungo, S. (2005). Establishing and structuring criteria for measuring knowledge management efforts. In *Proceedings of The 38th Annual Hawaii International Conference on System Sciences, 2005*

Argote, L. (2013). *Organizational learning: Creating, retaining and transferring knowledge*: Springer.

Bass, L., Clements, P., & Kazman, R. (2003). *Software architecture in practice*: Addison-Wesley Professional.

Beck, K. (2001). *Planning extreme programming*: Addison-Wesley Professional.

Berenson, S., Williams, L., & Slaten, K. (2005). Using Pair Programming and Agile Development Methods in a University Software Engineering Course to Develop a Model of Social Interactions. In *Proceedings of the International Conference on Crossing Cultures, Changing Lives*.

Billett, S., & Choy, S. (2013). Learning through work: emerging perspectives and new challenges. *Journal of Workplace Learning, 25*(4), 264-276.

Blech, J. O., & Glesner, S. (2004). A Formal Correctness Proof for Code Generation from SSA Form in Isabelle/HOL. *GI Jahrestagung (2), 51*, 449-458.

Brockmann, E. N., & Anthony, W. P. (2002). Tacit knowledge and strategic decision making. *Group & Organization Management, 27*(4), 436-455.

Cayaba, C., & Pablo, Z. (2013). A Qualitative Investigation of the SECI Model's Knowledge Conversions in the Applications Development Context.

Chau, T., & Maurer, F. (2004). Knowledge sharing in agile software teams *Logic versus approximation* (pp. 173-183): Springer.

Chigona, W., & Pollock, M. (2008). Pair programming for information systems students new to programming: Students' experiences and teachers' challenges. In *Proceedings of the International Conference on Management of Engineering & Technology, 2008*. PICMET 2008. Portland.

Chin, W. W. (1998). The partial least squares approach to structural equation modeling. *Modern methods for business research, 295*(2), 295-336.

Ciolkowski, M., & Schlemmer, M. (2002). Experiences with a case study on pair programming. In *Proceedings of the Workshop on Empirical Studies in Software Engineering*.

Cockburn, A. (2004). *Crystal clear: a human-powered methodology for small teams*: Pearson Education

Cockburn, A., & Williams, L. (2000). The costs and benefits of pair programming. *Extreme programming examined*, 223-247.

Cohen, L., Manion, L., & Morrison, K. (2000). Research Methods in Education [5 th edn] London: Routledge Falmer. *Teaching in Higher Education, 41*.

Crawford, B., de la Barra, C. L., Soto, R., Dorochesi, M., & Monfroy, E. (2013). The Role of Knowledge Management in Agile Software Development *HCI International 2013-Posters' Extended Abstracts* (pp. 17-21): Springer.

Dalkir, K. (2013). *Knowledge management in theory and practice*: Routledge.

de Azevedo Santos, M., de Souza Bermejo, P. H., de Oliveira, M. S., & Tonelli, A. O. (2011). Agile practices: An assessment of perception of value of professionals on the quality criteria in performance of projects. *Journal of Software Engineering and Applications, 4*(12), 700-709.

di Bella, E., Fronza, I., Phaphoom, N., Sillitti, A., Succi, G., & Vlasenko, J. (2013a). Pair Programming and Software Defects--A Large, Industrial Case Study. *Software Engineering, IEEE Transactions on, 39*(7), 930-953.

di Bella, E., Fronza, I., Phaphoom, N., Sillitti, A., Succi, G., & Vlasenko, J. (2013b). Pair Programming and Software Defects-A Large, Industrial Case Study.

Dijkstra, E. W. (1993). *A discipline of programming* (Vol. 1): prentice-hall Englewood Cliffs.

Dingsøyr, T., Nerur, S., Balijepally, V., & Moe, N. B. (2012). A decade of agile methodologies: Towards explaining agile software development. *Journal of Systems and Software, 85*(6), 1213-1221.

Dorairaj, S., Noble, J., & Malik, P. (2012). Knowledge Management in Distributed Agile Software Development. In *Proceedings of the Agile Conference* (AGILE), 2012.

Dudley, P. (2013). Teacher learning in Lesson Study: What interaction-level discourse analysis revealed about how teachers utilised imagination, tacit knowledge of teaching and fresh evidence of pupils learning, to develop practice knowledge and so enhance their pupils' learning. *Teaching and Teacher Education, 34*, 107-121.

Dybå, T., & Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review. *Information and software technology, 50*(9), 833-859.

Erdogmus, H. (2005). On the effectiveness of test-first approach to programming.

Fengjie, A., Fei, Q., & Xin, C. (2004). Knowledge sharing and web-based knowledge-sharing platform. In *Proceedings of IEEE International Conference on E-Commerce Technology for Dynamic E-Business*, 2004..

Fornell, C., & Bookstein, F. L. (1982). Two structural equation models: LISREL and PLS applied to consumer exit-voice theory. *Journal of Marketing Research (JMR), 19*(4).

Fornell, C., & Larcker, D. F. (1981). Evaluating structural equation models with unobservable variables and measurement error. *Journal of Marketing Research (JMR), 18*(1).

Foss, N. J., Lyngsie, J., & Zahra, S. A. (2013). The role of external knowledge sources and organizational design in the process of opportunity exploitation. *Strategic Management Journal*.

Gerard, J. G. (2003). Measuring knowledge source tacitness and explicitness: A comparison of paired items. In *Proceedings of the 5th Annual Organizational Learning and Knowledge Conference*.

Gerholm, T. (1990). On tacit knowledge in academia. *European Journal of Education*, 263-271.

Guthrie, S. (1996). The Role of Tacit Knowledge in Judgement and Decision Making.

Hannay, J. E., Arisholm, E., Engvik, H., & Sjoberg, D. I. (2010). Effects of personality on pair programming. *Software Engineering, IEEE Transactions on, 36*(1), 61-80.

Hannay, J. E., Arisholm, E., Engvik, H., & Sjoberg, D. I. K. (2010). Effects of Personality on Pair Programming. *IEEE Transactions on Software Engineering, 36*(1), 61-80. doi: http://dx.doi.org/10.1109/TSE.2009.41

Hashim, K. F. (2012). *Understanding the determinants of continuous knowledge sharing intention within business online communities.* AUT University.

He, W., Qi, C., Xu, X., & Guiyang, P. (2013). Linking Knowledge Sharing and Employee Creativity: Decomposing Knowledge Mode and Improving the Measure of Tacit Knowledge Sharing.

Henseler, J., Ringle, C. M., & Sinkovics, R. R. (2009). The use of partial least squares path modeling in international marketing. *Advances in international marketing*, *20*(1), 277-319.

Ho, C.-w. (2003). Tacit Knowledge Management and Pair Programming.

Ikujiro, N., & Takeuchi, H. (1995). The knowledge-creating company. *Harvard Business Review on Knowledge Management*.

Iranzo, P. J. (2002). On the correctness of the factoring transformation *Functional and Logic Programming* (pp. 119-133): Springer.

Jabar, M. A., Sidi, F., & Selamat, M. H. (2010). Tacit knowledge codification. *Journal of Computer Science, 6*(10), 1170.

Jensen, R. (2003). A pair programming experience. *The journal of defensive software engineering, 16*(3), 22-24.

Jiangping, W., Ming, Z., & Yahui, Z. (2013). Case Study on Tacit Knowledge Management Systems within X Company. *Technology and Investment, 4*, 92.

Kashefipour, S., Falconer, R., & Lin, B. (2002). Modeling longitudinal dispersion in natural channel flows using ANNs. *River Flow 2002*, 111-116.

Kashif, M., & Kelly, K. (2013). *Knowledge Management and Sharing Within Project Teams: A qualitative Study of Ericsson.* Mälardalen University.

Kavitha, R., & Ahmed, I. (2011). A Knowledge Management Framework for Agile Software Development Teams. In *Proceedings of the International Conference on Process Automation, Control and Computing (PACC),* 2011.

Keefe, K., Sheard, J., & Dick, M. (2006). Adopting XP practices for teaching object oriented programming. In *Proceedings of of the 8th Australasian Conference on Computing Education*-Volume 52.

Kent, A. E. S., & Selic, B. (2000). «UML» 2000–The Unified Modeling Language.

Kimble, C. (2013). What Cost Knowledge Management? The Example of Infosys. *Global Business and Organizational Excellence, 32*(3), 6-14.

King, W. R. (2009). *Knowledge management and organizational learning*: Springer.

Kline, R. B. (2005). Principles and Practice of Structural Equation Modeling. 2005. *New York, NY: Guilford*.

Koulikov, M. (2011). Emerging problems in knowledge sharing and the three new ethics of knowledge transfer. *Knowledge Management & E-Learning: An International Journal, 3*(2), 237--250.

Lavrakas, P. J. (2008). *Encyclopedia of survey research methods*: Sage.

Layman, L. (2006). Changing students' perceptions: An analysis of the supplementary benefits of collaborative software development. In *Proceedings of the 19th Conference on Software Engineering Education and Training,* 2006. Proceedings..

Leonard, N., & Insch, G. S. (2005). Tacit knowledge in academia: a proposed model and measurement scale. *The Journal of Psychology: Interdisciplinary and Applied, 139*(6), 495-512.

Lewis, B. R., Templeton, G. F., & Byrd, T. A. (2005). A methodology for construct development in MIS research. *European Journal of Information Systems, 14*(4), 388-400.

LI, Z.-x., WANG, Q., & Cao, L. (2013). An Analysis of the Structure and Evaluation Methods of Individual Tacit Knowledge.

Liao, S.-h., & Wu, C.-c. (2009). The relationship among knowledge management, organizational learning, and organizational performance. *International Journal of Business and Management, 4*(4), P64.

Liaw, S.-S., Huang, H.-M., & Chen, G.-D. (2007). Surveying instructor and learner attitudes toward e-learning. *Computers & Education, 49*(4), 1066-1080.

Lui, K. M., Barnes, K. A., & Chan, K. C. (2010). Pair Programming: Issues and Challenges *Agile Software Development* (pp. 143-163): Springer.

Mahdavi-Hezavehi, S., Galster, M., & Avgeriou, P. (2013). Variability in quality attributes of service-based software systems: A systematic literature review. *Information and software technology, 55*(2), 320-343.

Markoski, B., Hotomski, P., Malbaški, D., & Obradović, D. (2013). Resolution methods in proving the program correctness. *The Yugoslav Journal of Operations Research ISSN: 0354-0243 EISSN: 2334-6043, 17*(2).

Mazida, A. (2010). An investigation of knowledge creation processes in LMS-supported expository and PBL teaching methods. *Unpublished doctoral dissertation). Universiti Sains Malaysia*.

Md Rejab, M., Omar, M., & Ahmad, M. (2012). Knowledge internalization in pair programming practices. *Journal of Information and Communication Technology (JICT), 11*, 163-177.

Murphy, G., & Salomone, S. (2013). Using social media to facilitate knowledge transfer in complex engineering environments: a primer for educators. *European Journal of Engineering Education, 38*(1), 70-84.

Nissen, M. E. (2002). An extended model of knowledge-flow dynamics. *Communications of the Association for Information Systems, 8*(18), 251-266.

Nonaka, I., & Konno, N. (1998). The Concept of" Ba": Building A Foundation For Knowledge Creation. *California management review, 40*(3).

Nonaka, I., & Takeuchi, H. (1995). The knowledge-creating company. *1995*.

Nonaka, I., & Takeuchi, H. (1996). The knowledge-creating company: How Japanese companies create the dynamics of innovation. *Long Range Planning, 29*(4), 592.

Nunnally, J. C. (1994). Bernstein, IH (1994). Psychometric theory: New York: McGraw-Hill.

Okumus, F. (2013). Facilitating knowledge management through information technology in hospitality organizations. *Journal of Hospitality and Tourism Technology, 4*(1), 4-4.

Omar, M., Romli, R., & Hussain, A. (2007). Automated Tool to Assess Pair Programming Code quality.

Omar, M., Syed-Abdullah, S.-L., & Yasin, A. (2010) Adopting Agile Approach: A Case in Malaysia.

Porter, L., Guzdial, M., McDowell, C., & Simon, B. (2013). Success in introductory programming: what works? *Communications of the ACM, 56*(8), 34-36.

Rejab, M. M., Omar, M., & Mazida Ahmad, K. B. A. (2011). Pair Programming in Inducing Knowledge Sharing.

rey Voas, J. (1996). Testing software for characteristics other than correctness: Safety, failure tolerance, and security.

Rimington, K. B. (2010). Expanding the Horizons of Educational Pair Programming: A Methodological Review of Pair Programming in Computer Science Education Research.

Ryan, S., & O'Connor, R. V. (2013). Acquiring and Sharing Tacit Knowledge in Software Development Teams: An Empirical Study. *Information and software technology*.

Sajeev, A., & Datta, S. (2013). Introducing Programmers to Pair Programming: A Controlled Experiment *Agile Processes in Software Engineering and Extreme Programming* (pp. 31-45): Springer.

Salleh, N., Mendes, E., & Grundy, J. (2011). Empirical studies of pair programming for CS/SE teaching in higher education: A systematic literature review. *Software Engineering, IEEE Transactions on, 37*(4), 509-525.

Santos, V., Goldman, A., & Roriz Filho, H. (2013). The influence of practices adopted by agile coaching and training to foster interaction and knowledge sharing in organizational practices. In *Proceedings of the 46th Hawaii International Conference on System Sciences (HICSS)*, 2013.

Schultze, U., & Leidner, D. E. (2002). Studying knowledge management in information systems research: discourses and theoretical assumptions. *MIS quarterly*, 213-242.

Sharma, D. (2014). Knowledge Management & Organizational Structure: A study of indian companies. *International Journal of Organizational Behaviour & Management Perspectives, 2*(4), 585-593.

Sillitti, A., Succi, G., & Vlasenko, J. (2012). Understanding the impact of Pair Programming on Developers Attention. In *Proceedings of the ICSE2012,* Zurich, Switzerland.

Singh, A., Singh, K., & Sharma, N. (2013). Knowledge Management: the agile way. In *Proceedings of the Information and Knowledge Management*.

Sommerville, J., & Craig, N. (2012). *Implementing IT in construction*: Routledge.

SOUZA, C. R. D. (2012). Fostering Inter-Team Knowledge Sharing Effectiveness In Agile Software Development.

Stankosky, M. (2005). *Creating the discipline of knowledge management*: Routledge.

Straub, D., Boudreau, M.-C., & Gefen, D. (2004). Validation Guidelines For Is Positivist Research. *Communications of the Association for Information Systems, 13*.

Straus, S. E., Tetroe, J., & Graham, I. D. (2013). *Knowledge translation in health care: moving from evidence to practice*: John Wiley & Sons.

Sung, S. Y., & Choi, J. N. (2013). Do organizations spend wisely on employees? Effects of training and development investments on learning and innovation in organizations. *Journal of Organizational Behavior*.

Swamidurai, R. (2009). *Collaborative-adversarial pair (CAP) programming.* 3394643 Ph.D., Auburn University, Ann Arbor. Retrieved from http://eserv.uum.edu.my/docview/304850186?accountid=42599 ProQuest Dissertations & Theses Full Text database.

Syed-Abdullah, S. L., Omar, M., Hamid, M. N. A., bt Ismail, C. L., & Jusoff, K. (2009). Positive affects inducer on software quality. *Computer and Information Science, 2*(3), P64.

Tomayko, J. E. (2002). A comparison of pair programming to inspections for software defect reduction. *Computer Science Education, 12*(3), 213-222.

Urbach, N., & Ahlemann, F. (2010). Structural equation modeling in information systems research using partial least squares. *Journal of Information Technology Theory and Application, 11*(2), 5-40.

Wan, J., Wan, D., Luo, W., & Wan, X. (2011). Research on Explicit and Tacit Knowledge Interaction in Software Process Improvement Project. *JSEA, 4*(6), 335-344.

Wang, L. (2009). The multi-dimension spiral model of the knowledge-based corporation-theoretical thinking of beijing future advertising corporation. *Journal of Software, 4*(5), 469-477.

Werner, L., Denner, J., Campe, S., Ortiz, E., DeLay, D., Hartl, A. C., & Laursen, B. (2013). Pair programming for middle school students: does friendship influence academic outcomes? In *Proceedings of the 44th ACM technical symposium on Computer science education*.

Wiig, K. M. (2003). A knowledge model for situation-handling. *Journal of Knowledge Management, 7*(5), 6-24.

Williams, L. (2001). Integrating pair programming into a software development process. In *Proceedings of the 14th Conference on Software Engineering Education and Training*, 2001.

Williams, L. (2007). Lessons learned from seven years of pair programming at North Carolina State University. *ACM SIGCSE Bulletin, 39*(4), 79-83.

Williams, L. (2010). Pair Programming. *Encyclopedia of Software Engineering, 2*.

Williams, L., McCrickard, D. S., Layman, L., & Hussein, K. (2008). Eleven guidelines for implementing pair programming in the classroom. In *Proceedings of the Agile 2008 Conference*. AGILE'08..

Williams, L., Shukla, A., & Anton, A. I. (2004). An initial exploration of the relationship between pair programming and Brooks' law. In *Proceedings of the Agile Development Conference*, 2004.

Wohlin, C., Runeson, P., Hst, M., Ohlsson, M. C., Regnell, B., & Wessln, A. (2012). *Experimentation in software engineering*: Springer Publishing Company, Incorporated.

Wood, W. A., & Kleb, W. L. (2003). Exploring XP for scientific research. *Software, IEEE, 20*(3), 30-36.

Wray, S. (2010). How pair programming really works. *Software, IEEE, 27*(1), 50-55.

Xu, S., & Rajlich, V. (2006). Empirical validation of test-driven pair programming in game development. In *Proceedings of the 5th IEEE/ACIS International Conference on Computer and Information Science, 2006* and *2006 1st IEEE/ACIS International Workshop on Component-Based Software Engineering, Software Architecture and Reuse*. ICIS-COMSAR 2006..

Yi, J. (2006). Externalization of tacit knowledge in online environments. *International Journal on E-learning, 5*(4), 663-674.