UNIVERSITY OF GHANA

SOFTWARE ENGINEERING COMPETENCIES AND EMPLOYMENT OUTCOMES: EXAMINING THE MODERATING EFFECTS OF PROGRAMMING LANGUAGE PROFICIENCY AND AGE GROUP

BY

DAVID BOADI

(10933781)

A THESIS SUBMITTED TO THE DEPARTMENT OF OPERATIONS AND MANAGEMENT INFORMATION SYSTEMS, UNIVERSITY OF GHANA BUSINESS SCHOOL, UNIVERSITY OF GHANA, LEGON IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF AN MPHIL IN MANAGEMENT INFORMATION SYSTEMS DEGREE

NOVEMBER 2024

DECLARATION

I do hereby declare that this work is the result of my own research and has not been presented by anyone for any academic award in this or any other university. All references used in this work have been fully acknowledged.

I therefore bear responsibility for any shortcomings.

3rd November 2024

…… ……

DAVID BOADI DATE

(10933781)

4th November 2024

…… ……...

PROF. EMMANUEL AWUNI KOLOG DATE

(SUPERVISOR)

4th November 2024

…… ……...

DR. ACHEAMPONG OWUSU DATE

(CO-SUPERVISOR)

ABSTRACT

Software engineering is a constantly evolving field that is heavily reliant on the competencies of its personnel. In a developing country such as Ghana where demand for software engineering is on the rise due to efforts to digitalize the economy, there is a need for a comprehensive study on the competencies required by the software industry. The Vice President of Ghana, Dr. Mahamoud Bawumia's, recent efforts to digitalize the country's public sectors have sparked a surge of motivation among the youth, inspiring them to pursue careers in this field.

Although existing literature has investigated the competencies required by the software industry, a significant majority have been carried out in developed-country contexts. Hence, one of the driving factors of this research is the scarcity of SEC studies in Ghana and other developing countries. In addition, existing literature has not explored how these competencies affect employment outcomes. Hence, this study sought to understand the competencies that are required by software firms in Ghana as well as their influence on employment outcomes. Furthermore, the study explored the moderating effect of knowledge in programming languages and generational age on the relationship between competencies and employment.

To achieve the objectives of this study, a qualitative approach based on the interpretivism paradigm was utilized, using interviews as the primary data collection method. Through a semi-structured interview, data were collected from two software firms in Accra, Ghana. Both software firms had been operational for over 10 years, possessing the requisite staff strength and an extensive project portfolio. In all, six respondents, three from each firm participated in the study. The collected data were transcribed and analyzed thematically drawing from Miles and Huberman's (1994) thematic analysis approach.

The findings revealed 14 technical competencies and 14 soft skills that are in-demand by software firms in Ghana. The identified competencies, including proficiency in programming languages, database management, version control, security, problem solving, and communication, among others, largely aligned with existing literature. However, some of the identified competencies such as research and innovation, DevOps, understanding of the SDLCs, ethical hacking, and emotional intelligence among others were unique to this study. In addition, the study validated a conceptual framework based on the human capital theory (HCT) with the major competency dimensions: achievement orientation, domain knowledge, communication skills, problem-solving ability, professional work ethic ability, and teamwork ability as factors, employment as the outcome, and knowledge in programming languages and generational age as moderators. The study found all the competencies to influence employment outcomes though with varying degrees of influence.

Regarding the moderators, the study found knowledge in programming languages to influence the relationship between the factors - domain knowledge and problem-solving ability, and the outcome, which is employment. On generational age, it was found that the various generational cohorts exhibit differences in domain knowledge and problem-solving ability which in turn influences the employment of software engineers. However, the study found that whereas the various generations (X, Y, and Z) exhibit different communication skills, such differences do not influence the employment outcomes of software engineers.

In all, the study sheds light on key competencies that are required by Ghanaian software firms, the influence of competencies on employment, and the moderating influence of knowledge in programming languages and generational age. This study has implications for research, practice, and policy which is elaborated on in the study.

DEDICATION

To my inspirational mother Augustina Agyei, my late dad, and siblings.

ACKNOWLEDGEMENTS

I am thankful to the almighty God for granting me life and good health for me to complete this thesis.

I must concede that my pursuit of an MPhil Degree would not have been possible without the invaluable contributions of a number of persons. Firstly, I acknowledge and extend my enormous appreciation towards my supervisor, Prof. Emmanuel Awuni Kolog, and co-supervisor, Dr. Acheampong Owusu, for their inspiration and support throughout my endeavor to complete this thesis. Without their insightful comments and expert guidance, this thesis would not have seen the finish line. I extend my deepest gratitude to Dr. Sulemana Bankuoru Egala for his immense support, guidance, and inspiration.

My sincere gratitude to the two software firms who accepted my invitation to participate in this study: COLDSiS Ghana Limited (CGL) and the other who wish to remain anonymous. Their thoughtfulness, cooperation, and drive to contribute to the development of software engineering made it possible for this study to be completed. I am thankful to all my 2021/2022 coursemates for their support, encouragement, and comradeship throughout these difficult years.

Finally, I am very grateful to my mother, siblings, and Asem Wisdom Delali Kwame. Without their support and prayers, this would not have been possible.

TABLE OF CONTENTS

DECLARATION i
ABSTRACT ii
DEDICATION iv
ACKNOWLEDGEMENTS v
LIST OF FIGURES x
LIST OF TABLES xi
LIST OF ABBREVIATIONS xii
CHAPTER ONE 1
INTRODUCTION 1
1.1 Background of the Study 1
1.2 Problem Statement 3
1.3 Purpose of the Study 5
1.4 Research Objectives 5
1.5 Research Questions 6
1.6 Significance of the Study 6
1.7 Thesis Organization 7
CHAPTER TWO 9
LITERATURE REVIEW 9
2.1 Chapter Overview 9
2.2 Software Engineering 9
2.3 Software Engineering Process 10
2.3.1 Software Specification 11
2.3.2 Software Design and Implementation 11
2.3.4 Software Verification and Validation 12
2.3.5 Software Maintenance 13
2.4 Software Process Models 14
2.4.1 Structured Development 14
2.4.2 Rapid Application Development (RAD) 15
2.4.3 Agile Development 16
2.4.4 Object-Oriented Programming (OOP) 19
2.4.5 Summary of SDLC Models 20
2.5 Overview of the Software Engineering Landscape in Ghana 20
2.5.1 Categories of Software Engineering Firms in Ghana 22
2.5.2 Government Policies and Initiatives on Software Engineering 23

2.6 Generational Age and Employment 24
2.6.1 Gen X 25
2.6.2 Gen Y 25
2.6.3 Gen Z 26
2.6.4 Age Bias and Employment 26
2.7 Programming Languages and Employment 27
2.7.1 Overview of Programming Languages 27
2.7.2 Importance of Programming Language Knowledge in Software Engineering 28
2.7.3 Programming Language Knowledge and Employment 28
2.8 Software Engineering Competencies 29
2.9 Empirical Review of Existing Literature on SEC 30
2.9.1 Analysis of the Summarized SEC Literature 35
2.9.2 Research Gaps and Directions for Future Research 37
2.10 Chapter Summary 38
CHAPTER THREE 39
THEORETICAL FRAMEWORK 39
3.1 Chapter Overview 39
3.2 Employment Theoretical Models 39
3.2.1 Theory of Planned Behavior 40
3.2.2 Social Cognitive Career Theory 42
3.2.3 Human Capital Theory 43
3.3 Development of the Conceptual Framework 47
3.3.1 Achievement Orientation and Employment 47
3.3.2 Domain Knowledge and Employment 48
3.3.3 Communication Skills and Employment 49
3.3.4 Problem-solving Ability and Employment 50
3.3.5 Professional Work Ethic Ability and Employment 51
3.3.6 Teamwork Ability and Employment 52
3.3.7 Moderating Influence of Knowledge in Programming Languages 53
3.3.8 Moderating Influence of Generational Age 54
3.4 Chapter Summary 59
CHAPTER FOUR 60
RESEARCH METHODOLOGY 60
4.1 Chapter Overview 60
4.2 Research Paradigm 60

4.2.1 Positivism 61
4.2.2 Interpretivism 61
4.2.3 Critical Realism 61
4.3 Research Design and Methods 62
4.4 Case Study as a Research Method 64
4.4.1 Case Study Design 65
4.4.2 Sampling Technique for Selecting Respondents 66
4.4.3 Data Collection Methods 67
4.5 Data Analysis and Interpretation 70
4.5.1 Thematic Analysis 70
4.6 Chapter Summary 73
CHAPTER FIVE 74
RESEARCH FINDINGS 74
5.1 Chapter Overview 74
5.2 Case Description 74
5.2.1 SwF 1 74
5.2.2 Services Provided 75
5.2.3 SwF 2 75
5.2.4 Services Provided 76
5.3 Demographic Characteristics of Participants 76
5.4 Findings from the Case 79
5.4.1 Case one: Software Engineering Competencies (SEC) required by Employers in Ghana (SwF 1) 79
5.4.2 Achievement Orientation and Employment 85
5.4.3 Domain Knowledge and Employment 87
5.4.4 Communication Skills and Employment 89
5.4.5 Problem-Solving Ability and Employment 91
5.4.6 Professional Work Ethic Ability and Employment 92
5.4.7 Teamwork Ability and Employment 95
5.4.8 Impact of Knowledge in Programming Languages 97
5.4.9 Impact of Generational Age 100
5.4.10 Case two: Software Engineering Competencies (SEC) required by Employers in Ghana (SwF 2) 105
5.4.11 Achievement Orientation and Employment 111
5.4.12 Domain Knowledge and Employment 112

5.4.13 Communication Skills and Employment 114
5.4.14 Problem Solving Ability and Employment 115
5.4.15 Professional Work Ethic Ability and Employment 117
5.4.16 Teamwork Ability and Employment 118
5.4.17 Impact of Knowledge in Programming Languages 119
5.4.18 Impact of Generational Age 122
5.5 Chapter Summary 127
CHAPTER SIX 128
ANALYSIS AND DISCUSSION OF FINDINGS 128
6.1 Chapter Overview 128
6.2 Software Engineering Competencies required by Employers in Ghana 128
6.3 Influence of Software Engineering Competencies on Employment 131
6.4 Chapter Summary 153
CHAPTER SEVEN 154
SUMMARY, CONCLUSION AND RECOMMENDATIONS 154
7.1 Chapter Overview 154
7.2 Summary of the Research Process 154
7.3 Addressing the Research Objectives and Questions 159
7.4 Summary of the Research Findings 164
7.5 Conclusion 165
7.6 Implications of the Study 166
7.6.1 Implication to Research 166
7.6.2 Implication to Practice 167
7.6.3 Implication to Policy 167
7.7 Research Limitations 168
7.8 Future Research Directions 168
REFERENCES 170
APPENDICES 200
Appendix A – Research Interview Guide 200

LIST OF FIGURES

Figure 2.1 Fundamental Software Engineering Processes 10

Figure 3.1 Theory of Planned Behavior 41

Figure 3.2 Social Cognitive Career Theory (SCCT) 43

Figure 3.3 HCT Framework 44

Figure 3.4 Conceptual Framework based on the Human Capital Theory 58

Figure 4.1 Miles and Huberman's Data Analysis 71

Figure 4.2 Flowchart of the Research Process 72

Figure 7.1 Revised Conceptual Framework based on findings from Software Firms 157

LIST OF TABLES

Table 2.1 Breakdown of the estimated market value of Ghana's ICT sector 21

Table 2.2 Summary of SEC Literature 31

Table 5.1 Demographic Characteristics of Participants 77

Table 5.2 Software Engineering Competencies, Description, and Importance (Case 1) 79

Table 5.3 Software Engineering Competencies, Description and Importance (Case 2) 106

Table 6.1 Software Engineering Competencies required by Employers in Ghana 128

Table 6.2 Achievement Orientation Factor item 131

Table 6.3 Domain Knowledge Factor item 134

Table 6.4 Communication Factor item 136

Table 6.5 Problem-Solving Ability Factor item 138

Table 6.6 Communication Factor item 140

Table 6.7 Teamwork Ability Factor item 142

Table 6.8 Knowledge in Programming Languages and Domain Knowledge Factor item 143

Table 6.9 Knowledge in Programming Languages and Problem-Solving Ability Factor item 145

Table 6.10 Generational Age and Domain Knowledge Factor item 147

Table 6.11 Generational Age and Communication Factor item 150

Table 6.12 Generational Age and Problem Solving Ability Factor item 152

Table 7.1 Postulations made from findings 158

Table 7.2 Mapping out Study Objectives with Findings, Literature, and Contributions 160

LIST OF ABBREVIATIONS

SE	Software Engineering
SEC	Software Engineering Competencies
TPB	Theory of Planned Behavior
SCCT	Social Cognitive Career Theory
HCT	Human Capital Theory
SRS	Software Requirements Specification
V & V	Software verification and validation
SDLC	Software Development Life Cycle
RAD	Rapid application development
XP	Extreme Programming
WIP	Work in progress
OOP	Object-Oriented Programming
UML	Unified Modeling Language
ICT4AD	ICT for Accelerated Development
UI	User Interface
UX	User Experience
FinTech	Financial Technologies
CRM	Client Relationship Management
HRMS	Human Resource Management Systems
IoT	Internet of Things
AI	Artificial Intelligence
TDD	Test-Driven Development
QA	Quality Assurance
UAT	Unit Aceeptance Testing

CHAPTER ONE

INTRODUCTION

1.1 Background of the Study

One industry that holds much promise regarding employment for graduates is that of software engineering (SE). In today's era of digital transformation, organizations are continually seeking to leverage technological advancements to be innovative and efficient (Allioui & Mourdi, 2023). As a result, the demand for SE professionals has significantly increased (Zhang & Gu, 2023). According to the U.S. Bureau of Labor Statistics 2023, SE jobs are expected to grow by 25 percent between 2021 and 2031 surpassing the national average growth rate for all professions of 8 percent. Furthermore, Google's Africa Developer Ecosystem Report for 2021 indicates a rising demand for African SE professionals, resulting in a 3.8 percent expansion of the SE workforce, despite prevailing economic contractions (Gajria, 2022).

To gain employment, however, one needs to possess competencies that are in demand by the respective industry. This is particularly significant in the SE industry, as it is heavily reliant on the competencies of its professionals, unlike other fields (Assyne et al., 2022). In the realm of software engineering, two major classifications of competencies have been identified: technical (hard) and soft competencies (Jebreen & Nabot, 2021; Ntinda et al., 2021). While technical competencies are often measurable and are mostly acquired through education and training, soft competencies such as teamwork and communication skills are less measurable and are often considered to be personality traits (Mangiza & Brown, 2020; Rabelo et al., 2022). Software engineers are thus required to possess a good blend of both technical and soft competencies to stay relevant in the industry (Ntinda et al., 2021). However, it is imperative to understand the specific hard and soft competencies that are in demand and the impact of those competencies on employment in the industry.

Additionally, the link between SEC and employment is not always straightforward; it is often complex and influenced by numerous factors (Monteiro et al., 2016). Two such factors are knowledge in programming languages and generational age (Gabbrielli & Martini, 2023; Urick, 2017). Programming languages such as C++, Python, and Java are considered fundamental in software development as they provide the medium through which instructions are given to the computer to execute (Abdulkareem & Abboud, 2021; Ali & Qayyum, 2021). Without them, software applications such as websites, games, and mobile applications among others cannot be developed (Gabbrielli & Martini, 2023). Hence, despite the recent emergence of the citizen developer/low and no code development, knowledge in programming languages remains a vital asset in the software engineer's toolbox (Lebens et al., 2021). However, as software professionals equip themselves with relevant programming languages, it is necessary to investigate their impact on employment.

Also, generational age is a factor that can influence the hiring decisions of organizations, including those in the software industry. Existing literature identifies three major generational cohorts namely Gen X (born 1965 - 1981), Gen Y also known as Millennials (born 1982 - 1999), and Gen Z (born 2000 - 2012) (Mahmoud et al., 2021). Each of these generational cohorts shares distinct beliefs, values, and characteristics that influence their learning styles and work attitudes. For instance, whereas Gen X are more independent and values work-life balance, Gen Y considered the first digital natives, are known to be tech-savvy, confident, and adaptable (Maan & Srivastava, 2023; Mahmoud et al., 2021; Nelson & Duxbury, 2021). Gen Z, also known as the 'Facebook generation' are considered the real digital natives since they were born and raised in a world of technology (Maan & Srivastava, 2023). They form the youngest workforce, and are considered to be creative and prefer the usage of digital platforms

for communication (Borg et al., 2023; Egerová et al., 2021). Although the differing worldview of the various generational cohorts presents a diverse range of skills to organizations (Borg et al., 2023), it is important to understand the extent to which they impact the employment of professionals in the SE industry.

Given the points mentioned above, this study therefore proceeds to explore the competencies required in SE, their impact on employment, and the moderating influence of knowledge in programming languages and generational age on the relationship between SEC and employment. Thus, the study proposes a moderated model in which the employment of software engineers in the industry depends on the competencies they possess, moderated by their knowledge in programming languages and the generational cohort they belong.

1.2 Problem Statement

SE projects have been found to have relatively low success rates compared to other types of projects (Kusumasari et al., 2018). A major recurring theme to these failures is often attributed to staff competence (Hasan et al., 2023; Lehtinen et al., 2014; Mandal & Pal, 2015). As such, the competency of software professionals has become a significant subject, as the SE industry is heavily reliant on the knowledge, skills, and abilities of its professionals (Garousi et al., 2020; Tam et al., 2020). Several studies (Jia et al., 2017; Moustroufas et al., 2015; Sedelmaier & Landes, 2014) have therefore proposed models that standardize SE competencies to ensure effective and structured assessment of professionals in the field. However, the impact of those competencies on employment has not been explored by existing literature. Also, the impact of knowledge in programming languages and generational age on the employment of software engineers has not been explored by existing literature.

In addition, reviewed literature suggests that limited studies on SEC have been carried out in developing country contexts. This includes a study conducted by Ntinda et al. (2021) where the authors explored the competencies that are expected of SE graduates by the software industry in Namibia. Another study by Marebane and Hans (2021) analyses the content of the curriculum of six Technology Universities in South Africa unraveling the ethics competency gap in the process. Although significant studies on SEC carried out in developed country contexts offer valuable insights (Galster et al., 2022; Groeneveld et al., 2020; Stamm, 2023), attributes such as culture, technology, and education differ from developing country contexts such as Ghana. Hence, the limited research on SEC in the African and developing country context underscores the need for more in-depth exploration of the subject in the region.

Furthermore, studies on generational age mostly employ quantitative methods to offer statistical insights into the differences that exist among the various cohorts in terms of work values, attitudes, and learning styles among other factors (Maan & Srivastava, 2023; Magni & Manzoni, 2020; Standifer & Lester, 2020). However, employing a qualitative approach to explore the influence of SEC on employment outcomes across different generational cohorts could offer deeper and more valuable insights. Qualitative methods, such as interviews and thematic analysis, can provide a nuanced understanding of how SEC impacts employment outcomes across different generations. This approach serves to offer new perspectives while enriching the already existing body of knowledge on generational age.

Lastly, research studies on employability, including those in the SE field, have been criticized for lacking theoretical underpinnings (Forrier et al., 2018). Several studies have focused on software development frameworks emphasizing the methodologies used in developing software (Fauzi et al., 2023; Mishra & Alzoubi, 2023; Nugroho et al., 2017). However, a major

area of information systems (IS) research that these studies frequently ignore is the socio-technical frameworks that examine the interaction between social and technical elements in the workplace. Yet, the socio-technical dimension in contemporary SE programming endeavors remains (Hoda, 2022; Storey et al., 2020). However, a preliminary review of existing literature suggests the non-existence of the application of theory in SEC studies. Socio-technical frameworks aid in understanding the intricacies of IS research by integrating human factors, organizational dynamics, and technological components (Ciriello et al., 2024; Ghaffarian, 2011). This trajectory has however been given less attention in existing IS literature. Hence, this study seeks to fill the gaps in existing research by applying the human capital theory in exploring how generational age and knowledge in programming languages moderate the relationship between SEC and employment in Ghana. This will be achieved through a qualitative approach to gain deeper insights into the phenomenon under investigation.

1.3 Purpose of the Study

The purpose of this thesis is to evaluate the impact of software engineering competencies on employment while also exploring how generational age and proficiency in programming languages influence this relationship.

1.4 Research Objectives

RO1: To explore software engineering competencies required by employers in Ghana.

RO2: To explore the impact of software engineering competencies on employment in Ghana.

RO3: To investigate the moderating influence of generational age and knowledge in programming languages on software engineering competencies and employment.

1.5 Research Questions

RQ1: What are the specific software engineering competencies required by employers in Ghana?

RQ2: How do software engineering competencies influence employment opportunities in Ghana?

RQ3: How do generational age and knowledge in different programming languages influence the relationship between software engineering competencies and employment?

1.6 Significance of the Study

The significance of the research is in three strands: knowledge, practice, and policy. Concerning knowledge, this study examines the influence of software engineering competencies on employment using a qualitative approach. Moreover, the study explores the moderating influence of knowledge in programming languages and generational age on the relationship between SEC and employment. This is a novel contribution to the research domain of SEC and will inform future studies, particularly in developing countries.

In terms of practice, the study will inform software professionals of the specific competencies required by the SE industry. This will lead to the development of essential competencies, which in turn will improve software quality.

With regard to policy, findings from the study will guide the development of curriculum and certification programs for SE professionals. Additionally, the study will shed light on potential age bias in hiring practices, offering insights for policymakers to promote age-neutral recruitment initiatives.

1.7 Thesis Organization

The study has been divided into seven (7) chapters, in accordance with the procedures of this research.

Chapter one provides an overview of the research study. This was achieved by giving an account of the background of the study, establishing the research problem that led to the study, and posing the relevant questions the study seeks to answer.

Chapter two provides a detailed review of relevant literature on SEC, employment, generational age, and knowledge in programming languages. The chapter also reviews relevant SE concepts and offers an overview of the SE landscape in Ghana.

Chapter three provides a comprehensive account of the theoretical foundation of the study. It discusses the human capital theory that underpins this research, leading to the creation of the conceptual framework of the study.

Chapter four provides a detailed account of the research paradigm and research methodology used for the study as well as their philosophical justifications. In this chapter, the methods used for data collection, sampling techniques, and the instruments employed, are presented and discussed.

Chapter five is dedicated to presenting the research findings, with a focus on the data presentation.

Chapter six analyzes the findings of the study and discusses them in relation to the propositions framed for the study.

Chapter seven summarizes and concludes the entire research. In this chapter, the research findings are synthesized, conclusions are drawn to address the research questions, and insights are provided to benefit academia, the SE industry, and other relevant stakeholders.

CHAPTER TWO

LITERATURE REVIEW

2.1 Chapter Overview

As highlighted in the introductory chapter, while there are existing studies on SEC, there is a need to explore the impact of SEC on the employment of software professionals. This need is driven by the industry's increasing significance in the employment of Ghanaian SE professionals. This chapter takes a step towards fulfilling this objective by reviewing previous research on SEC, the existing software engineering landscape in Ghana, and the conceptual approaches that underpin studies in this field. The aim is to provide a clear understanding of the SEC research domain and identify gaps in existing research while providing a solid foundation for further exploration of the study.

2.2 Software Engineering

Software engineering (SE) is a field that has been described in various ways in scholarly literature, reflecting its multifaceted function in both academic and industrial contexts. Sommerville (2011) describes SE as "an engineering discipline that is concerned with all aspects of software production" (p. 6). According to Sommerville, SE encompasses not only the developed software itself but also all accompanying documentation and configuration data required for the software to function correctly. Similarly, the IEEE defines SE as "the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software" (Bourque & Fairley, 2014, p. xxxi). Mall (2018) offers another perspective, considering SE as a systematic collection of effective program development practices and techniques, informed by both research innovations and the accumulated experience of programmers over the years. Moreover, Sarker et al. (2015) defines SE as "designing, writing, testing, implementing and maintaining software" (p. 61), emphasizing that

SE is integral to both the design and operationalization of computer systems and foundational in enabling the core functionalities that power modern computerized platforms. For the purpose of this research, a synthesized definition adapted from both Sommerville (2011) and the IEEE (Bourque & Fairley, 2014) will be utilized. Software engineering is defined for this study as "the application of a disciplined and systematic approach to all aspects of software production, encompassing development, operation, maintenance, as well as all accompanying documentation and configuration data".

2.3 Software Engineering Process

In light of the earlier discussion of the definitions and scope of SE, it becomes necessary to delve into the fundamental activities that characterize the field. According to Sommerville (2011), there are four fundamental SE activities: software specification (requirements engineering), software development (design and implementation), software verification and validation, and software evolution (maintenance). The understanding of these fundamental activities is crucial as they serve as the backbone of any SE project, outlining the lifecycle and deliverables at each stage (Mall, 2018). Collectively, these processes are often referred to as the Software Development Life Cycle (SDLC) (Sarker et al., 2015). Figure 2.1 depicts the fundamental SE Processes.

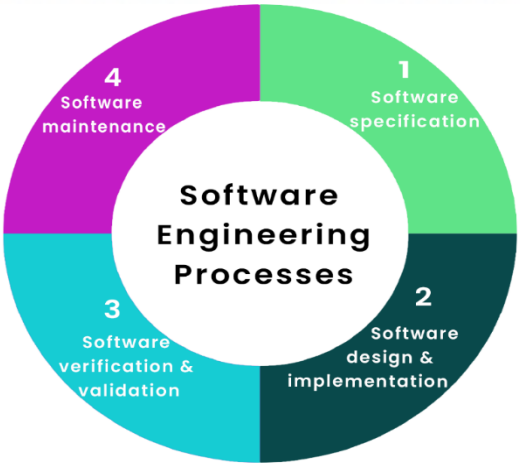

Figure 2.1 Fundamental Software Engineering Processes

2.3.1 Software Specification

Software can be very complex systems that require painstaking consultation with clients and relevant stakeholders to elicit their expectations before software development (Maalej et al., 2019). This requires the software engineer to elicit the functionality and constraints that a client desires for a software system, as well as document and manage these user requirements throughout the software development life cycle (Zhi & Morisaki, 2021). This SE activity is vital in helping avoid software crises by ensuring that developed software meets the specific needs and expectations of clients and relevant stakeholders.

There are three major steps to software specification; gathering requirements from clients or sponsors, documenting the gathered requirements, and reviewing the requirements to make sure they reflect the needs of the client (Al-Msie'Deen et al., 2021). The product of this SE activity is Software Requirements Specifications (SRS), a document that gives a detailed description of a software system slated for development (Zhi & Morisaki, 2021). SRS specifies the key features, constraints, and functionalities of a software product. The SRS document should be clearly understood and agreed upon by all stakeholders such as users, analysts, and developers (Kici et al., 2021). SRS is considered the most important artifact of the software development process (Darnoto & Siahaan, 2021), as it serves as an agreement between customers and developers, provides a basis for estimating costs and timelines and serves as a reference point for validation and verification processes (Mall, 2018).

2.3.2 Software Design and Implementation

The next activity after software specification is the design and implementation of the software system. This stage involves designing the software architecture, developing algorithms, and writing code to implement the software system (Mall, 2018). The main objective of the design

phase is to create a design document based on the Software Requirements Specification (SRS). The design document includes elements such as module design, module interfaces, data structures within modules, and the algorithms required for each module (Mall, 2018). In essence, the software design process transforms the SRS into a practical guide for subsequent implementation. During the implementation stage, developers use programming languages and development tools to actualize what has been specified in the design document (Sommerville, 2011).

2.3.4 Software Verification and Validation

Verification and validation (V & V) are two important activities carried out to ensure software conforms to their specifications and that customer requirements are satisfied (Sommerville, 2011). Verification is primarily focused on ensuring that each step of the software development is carried out accurately. On the other hand, validation, which takes place towards the end of the development process, aims to determine whether the right product has been developed (Mall, 2018). During V & V, different types of tests are conducted on the software system, which includes unit testing, integration testing, system testing, and acceptance testing (Jan et al., 2016).

Unit testing involves testing individual components or units of a software program in isolation and is often performed by developers themselves. The objective is to ensure that each unit of the software performs as designed (Mall, 2018). Integration testing is carried out after unit testing and requires the individual modules to be combined and tested as a group. This type of testing is designed to detect faults in the interaction between integrated units (Laplante, 2007). System testing involves testing the entire software system to confirm its compliance with the specified requirements. The main objective is to examine whether all components of the system

interact as expected and whether the system functions as intended (Sommerville, 2011). It is considered a high-level test that evaluates the overall performance of the system. Acceptance testing is conducted by the customer before the software is launched. Its main purpose is to allow the customer to validate that the software fulfills all the pre-established requirements and is ready for actual use (Mall, 2018). V & V ensures that the software is free from errors and faults before it is deployed for use by the client or sponsor.

2.3.5 Software Maintenance

After the developed and tested software is delivered to clients or sponsors, software maintenance is carried out to make modifications to the software to meet changing customer needs (Sommerville, 2011). Software maintenance is estimated to account for 70% of overall software costs, highlighting its significance within the software development life cycle (Ogheneovo, 2014; Venters et al., 2018). There exist three major types of software maintenance: corrective, adaptive, and perfective (Ali et al., 2020; Heričko et al., 2022; Mall, 2018). Corrective maintenance seeks to fix any errors that are identified while clients are using the software (Ali et al., 2020; Chen et al., 2021; Mall, 2018). Adaptive maintenance is carried out to allow the software to accommodate new environments such as changes in the operating system or to allow seamless integration with new software platforms (Ali et al., 2020; Chen et al., 2021; Mall, 2018). Perfective maintenance involves adding new features, updating documentation, and improving the overall efficiency and performance of the delivered software (Ali et al., 2020; Chen et al., 2021; Mall, 2018). The significance of software maintenance cannot be overstated, though it is often overshadowed by initial development (Mall, 2018). Due to the potentially high associated costs, it is imperative for organizations to understand and manage software maintenance effectively.

2.4 Software Process Models

In exploring the realm of SE, it is essential to understand the various SDLC models that guide the development and maintenance of software systems. Software process models also known as SDLC models are frameworks that provide a structured approach for developing software products (Sarker et al., 2015; Sommerville, 2011). These models provide a structured and organized way to manage and control the software development process from initiation to deployment and maintenance. There exist a variety of SDLC models and each comes with an approach towards project workflow and has particular strategies for addressing project risks and costs (Sarker et al., 2015). The SDLC models can be grouped according to their primary related methodologies: structured development, rapid application development, agile development, and object-oriented programming. This organization aids in a coherent understanding of each model within a broader context.

2.4.1 Structured Development

Structured development methodologies emphasize detailed planning, documentation, and process management (Estler et al., 2014). They are inflexible after the requirements-gathering stage, making any subsequent design changes possible only in future version releases (Estler et al., 2014; Mishra & Alzoubi, 2023; Papadopoulos, 2015). These linear, phase-specific approaches offer proof of functional software late in the project (Estler et al., 2014). They include the Waterfall model, the V-shape model, and the Parallel model.

2.4.1.1 Waterfall Model

The Waterfall model, which is widely used in software development, adopts a linear approach where each phase must be fully completed before proceeding to the next (Sarker et al., 2015). These phases generally encompass requirements gathering, system design, implementation,

testing, deployment and maintenance. The Waterfall model, which is arguably the most intuitive way of developing software, was very popular in the 1970s and is still used across many software projects (Mall, 2018). This model is well-suited for projects with stable and well-defined requirements (Kaur & Sengupta, 2013). Implementation of the waterfall model involves strong planning and documentation skills (Adel & Abdullah, 2015), thorough understanding of software requirements (Sommerville, 2011), ability to work in a structured and organized manner (Adel & Abdullah, 2015), and proficiency in design and coding standards (Sarker et al., 2015).

2.4.1.2 Parallel Development Model

The Parallel Development model allows for simultaneous progress on components of a software project, unlike linear models such as the Waterfall model that require the completion of one stage before proceeding to the next stage (Nugroho et al., 2017). Parallel Development deviates from the waterfall model by breaking down the project into sub-projects that can be worked on concurrently (Yu, 2018). This leads to faster development times and provides more flexibility during the design and implementation phases, addressing the limitations of sequential development. The final integration takes place after all sub-projects have been completed and refined, helping to expedite software delivery and effectively manage complexities in software development projects (Nugroho et al., 2017; Yu, 2018).

2.4.2 Rapid Application Development (RAD)

Rapid application development (RAD) was introduced in 1991 to overcome the rigidity of structured approaches such as the waterfall model, which is less accommodating to change requests from clients (Fauzi et al., 2023). RAD places less emphasis on planning, rather it focuses on rapid releases of prototypes based on given requirements and making multiple

iterations until the client is satisfied (Sarker et al., 2015). The RAD Model is founded on the premise that users can provide more valuable feedback when they directly interact with a live system (Sarker et al., 2015; Sommerville, 2011).

2.4.2.1 Prototyping

Prototyping is a RAD approach which focuses on creating an initial working model or prototype of the system early in the project lifecycle instead of relying on complete system specifications from the start (Mall, 2018). The main goal is to develop a functional though not complete version of the application to showcase its concept, design and functionality. This prototype is then shared with stakeholders and end users for their feedback. Based on user input, the prototype is then modified, improved or even completely reworked (Sommerville, 2011). This iterative process continues until the system adequately meets users needs and expectations. Prototyping aids in removing any ambiguities or misunderstandings early in the project's life cycle, thereby helping to reduce costly changes later in development and promotes an efficient development process that prioritizes user satisfaction (Sarker et al., 2015).

2.4.3 Agile Development

The Agile methodology involves dividing a project into phases and emphasizes constant collaboration with stakeholders, ensuring continuous improvement throughout each stage (Sommerville, 2011). Teams follow a cycle of planning, executing, and evaluating, placing great importance on ongoing collaboration with both team members and project stakeholders. The Agile methodology emerged as a response to the rigid and linear nature of traditional methodologies like the Waterfall Model (Sarker et al., 2015). This methodology emphasizes working software over comprehensive documentation (López-Alcarria et al., 2019).

Agile is centered on embracing change and delivering functional software rapidly, while continuously enhancing the product based on customer feedback and evolving requirements. The inclusion of a designated customer representative (CR) plays a vital role in ensuring that agile methods generate software functionality that closely aligns with customer requirements (Matook & Maruping, 2014). Agile emphasizes good communication skills, teamwork, adaptability, collaboration with clients, and familiarity with Test-Driven Development (TDD), continuous integration, and pair programming (Conforto et al., 2014; Kropp et al., 2016; López-Alcarria et al., 2019). This approach encompasses various well-known methodologies, such as Kanban, Scrum, and Extreme Programming (XP), each having its distinct characteristics and practices.

2.4.3.1 Scrum

Scrum is a widely used project management framework in software development (Srivastava et al., 2017). It is recognized for its incremental approach, which enables adaptability and swift responses to changes. In Scrum, work is segmented into manageable portions known as "sprints", typically spanning one to four weeks (Gonçalves, 2018). Each sprint commences with a planning meeting, during which the team determines the tasks to be completed within that period. The team then embarks on the tasks assigned for the sprint, and upon completion, they review the work as well as any challenges encountered along the way.

Key roles in Scrum include the Product Owner, who represents stakeholders and business interests; the Scrum Master, who facilitates the process and resolves team issues; and the Development Team, who are entrusted with executing the assigned work (Gonçalves, 2018). Scrum places a strong emphasis on collaboration, functional software delivery, team autonomy, and adaptability to evolving business circumstances (Rola et al., 2016). As a result, it has

gained popularity among teams seeking a flexible approach to managing projects amidst ever-changing requirements.

2.4.3.2 Kanban

Kanban is an agile methodology that places emphasis on visualizing the workflow, constraining work in progress (WIP), and driving continuous process improvement (Alaidaros et al., 2021). It employs a visual board known as a Kanban board, which is divided into columns representing distinct stages of the development process (Saltz & Heckman, 2020). Tasks are visualized as cards and are moved across the columns as they advance, enabling easy monitoring of workflow and identification of bottlenecks. By restricting WIP, Kanban encourages the team to concentrate on completing tasks before commencing new ones, mitigating the risks of multitasking and enhancing overall efficiency (Alaidaros et al., 2021).

2.4.3.3 Extreme Programming (XP)

Extreme Programming (XP) is an agile methodology that places strong emphasis on continuous feedback, active customer involvement, and the delivery of high-quality code (Akhtar et al., 2022). It introduces a set of engineering practices, including Test-Driven Development (TDD), Continuous Integration, Pair Programming, and Refactoring, to enhance software quality and development efficiency (Duncan, 2015). XP fosters regular communication and close collaboration between developers and customers, as well as within the development team. The methodology upholds the concept of "embracing change", making it easier to incorporate new requirements and adjustments at any stage of the development process (Sohaib et al., 2019).

2.4.4 Object-Oriented Programming (OOP)

Object-Oriented Programming (OOP) is an approach, to software development that utilizes the concepts and principles of Object-Oriented Programming (OOP). Unlike structured methodologies that focus on task sequences, OOP places emphasis on the objects involved in a situation and the actions that can be performed on them (Mall, 2018). The methodology starts with problem identification and analyzing requirements where key entities (objects) and their interactions (methods) are recognized. These entities are then represented as classes, which act as blueprints for creating objects. Classes encapsulate data and behaviors promoting reusable code (Herrity, 2023). OOP methodology often adopts Unified Modeling Language (UML) to visually depict relationships among classes. UML diagrams serve as a roadmap for developers outlining system architecture before coding.

The four fundamental principles of OOP: encapsulation, inheritance, polymorphism, and abstraction, function not only as programming concepts but also guide this methodology (Herrity, 2023). For instance, encapsulation enhances data integrity and protection by restricting access or modification. Inheritance facilitates a structure where shared functionalities reside in parent classes resulting in maintainable and extensible code. Polymorphism allows for the use of an interface, with data types making code more flexible and easier to comprehend while abstraction reduces complexity by concealing the workings of classes and displaying only the essential functionalities (Herrity, 2023; Mall, 2018). OOP methodologies have given rise to the creation of programming languages that follow an object-oriented approach, such as the widely recognized C++ and Java (Doherty, 2023). In general, Object-Oriented Programming methodology offers a robust framework for software development, enabling teams to create maintainable, scalable, and high-quality software.

2.4.5 Summary of SDLC Models

In summary, Agile has emerged as the predominant approach, encompassing various methodologies like Scrum, Kanban, and Extreme Programming (Baxter & Turner, 2021; Veiga, 2017). These methodologies share core values and principles, including flexibility, collaboration, and iterative development. By empowering teams to swiftly deliver top-notch software, adapt to changing requirements, and consistently improve their processes, Agile methodologies prove exceptionally well-suited for the dynamic and fast-paced nature of modern development environments. Meanwhile, the Waterfall model retains its relevance for smaller projects with well-defined requirements (Andrei et al., 2019).

Recently, there has been a growing interest in the DevOps approach. While DevOps is not a traditional software process model, it seeks to connect development and operations promoting an integrated culture that improves the software delivery process (Ebert & Hochstein, 2023). This approach fosters a culture of integration that enhances the software delivery process and aligns well with Agile principles, thereby boosting the efficiency and effectiveness of the software development life cycle (Mohammad, 2018). It is worth noting that the successful implementation of these models, including DevOps, requires specific skills and capabilities. Hence, it is vital for software engineers to grasp and nurture these competencies to navigate and thrive in these development environments.

2.5 Overview of the Software Engineering Landscape in Ghana

Ghana's software engineering landscape, though relatively new, has seen a remarkable rise in recent times. This is due to the country's growing embrace of digitalization and automation. Domestic software development in the country has a lengthy history, dating back to the late 1980s with the like of SOFTtribe Limited (SOFT) pioneering commercialization of software

(Ahiabenu, 2017). Presently, Ghana serves as a thriving ecosystem for software companies, offering software solutions ranging from mobile applications to enterprise software. The prominent rise of the Ghanaian SE sector is further evident in the emergence of Ghanaian software players like SOFTtribe, mPedigree, Rancard, Logiciel, and Nosmay, who are making waves globally with application platforms across various sectors, from finance and agriculture to healthcare (World Bank Group, 2019).

In Ghana, software development is a subset of the ICT/Telecommunications sector. This larger sector, ICT/Telecommunications, comprises four sub-sectors: Telecommunications, Broadcasting, Support Services, and Software Development (National Communications Authority, 2017). The estimated worth of the country's ICT sector is currently valued at $1 billion. This figure is anticipated to grow to $5 billion by 2030 due to significant investments made by private entities and the government (Ankiilu, 2022). Table 2.1 gives a breakdown of the current value of Ghana's ICT sector.

Table 2.1 Breakdown of the estimated market value of Ghana's ICT sector

ICT infrastructure	$400 million
Software Sales	$200 million
Cloud infrastructure	$15 million
Cybersecurity	$30 million
Fintech, Health-tech and Ed-tech	$115 million
ICT Training & Support	$150 million

Source: International Trade Administration (2022)

It can be observed from Table 2.1 that Ghana has a sizable market for software that can be exploited by indigenous software companies. The large market size also presents employment opportunities for local software professionals. However, global giants like Oracle and Microsoft dominate the country's software market (International Trade Administration, 2022). Industry analysts have identified several factors contributing to this state of affairs, such as a shortage of trained workers, inadequate government policies in support of local software development, and the absence of a software development hub among others (Ahiabenu, 2017).

2.5.1 Categories of Software Engineering Firms in Ghana

Software development (software engineering) companies in Ghana fall into five different categories (National Communications Authority, 2017). Among them are:

Bespoke companies: This category of software firms develops custom software at the request of clients to meet specific user requirements. Besides, these firms develop software solutions as determined by market research to meet the demands and expectations of general software users. Thus, Bespoke companies provide custom-designed software to enhance and bridge specific gaps within their clients' business processes (National Communications Authority, 2017).

Consulting companies: These firms specialize in developing software applications such as mobile ad web for organizations in return for a fee. The development process can either be done in-house by their development teams or outsourced to external partners. Irrespective of the development approach, consulting firms maintain oversight over the application's life cycle (National Communications Authority, 2017).

Retailers: These companies serve as distributors and sales agents for established global software companies. Rather than developing software for clients, they represent internationally

recognized software firms to aid in the distribution and maintenance of already developed software. A substantial number of software developers in Ghana work for firms under this category (National Communications Authority, 2017).

Sector focused/specific service oriented: These firms concentrate on particular development sectors and usually remain within their chosen niches. They employ their own engineering teams who are responsible for designing, developing, debugging, managing, and updating their software. Many of these companies further specialize by creating applications that are tailored to specific sectors such as gaming, healthcare, and payment systems among others (National Communications Authority, 2017).

Website development companies: Prominent among the Ghanaian SE landscape are firms that specialize in the development of diverse kinds of websites for clients ranging from static to interactive websites. This is achieved with the aid of front-end and scripting languages as well as popular CMS platforms such as Joomla and WordPress which enable clients to easily manage and edit their own content (National Communications Authority, 2017).

2.5.2 Government Policies and Initiatives on Software Engineering

The technology sector in Ghana has seen a rise in government backing in recent times with the introduction of programs like ICT for Accelerated Development (ICT4AD) and the National Telecom Policy (NTP) (National Communications Authority, 2017). These government initiatives are intended to stimulate innovation and growth in the sector.

The ICT4AD 2003 policy, for instance, has a wide range of ambitious objectives with the ultimate goal of transforming the country into a technology-driven economy and society.

Among its primary goals, the policy seeks to modernize critical sectors such as agriculture, education, health, and business through ICT integration to enhance efficiency and productivity. In addition, the policy seeks to bridge the digital divide, foster a knowledge-based economy, and improve public sector efficiency through the use of ICT. These government initiatives serve to expand the SE market, providing more employment opportunities for indigenous software developers.

2.6 Generational Age and Employment

The concept of generational age has received much attention in scholarly literature. The concept of generational age was first introduced into academic and public discourse in 1952 by a Hungarian sociologist, Karl Mannheim, in his essay "The Problem of Generations" (McCourt, 2012). The author argued that generations go beyond a group of persons born around the same time but a collective who share certain experiences, values, and attitudes from their formative years (Maan & Srivastava, 2023; Magni & Manzoni, 2020). William Strauss and Neil Howe later popularized the concept of generational age through their groundbreaking book "Generations" published in 1991 (Parry & Urwin, 2011). The influence of the theory cuts across disciplines such as sociology, marketing, and organizational studies. The theory helps in understanding the behaviors and predispositions of various generations which aids in explaining consumer behavior, political shifts, and workplace interactions (Cogin, 2012; Smith et al., 2019; Yawson & Yamoah, 2020). Currently, four generational cohorts exist in the workplace: Baby boomers, Gen X, Gen Y, and Gen Z (Ratajczak, 2020). Due to confusions that exist in literature concerning the cut-off points for birth years of the various generational cohorts, this study adopts the generational categorization used by Mahmoud et al. (2021). Gen X born between 1965 and 1981, Gen Y between 1982 and 1999, and Gen Z born between 2000 and 2012. The characteristics of the three generational cohorts are explained below.

2.6.1 Gen X

The Gen X cohort, born between 1965 and 1981, grew up during the beginning of technological advancements and economic transformation (Mahmoud et al., 2021). This generational cohort experienced firsthand the emergence of computers, the event of the Berlin Walls collapse and the introduction of the internet (Safi, 2019; Talarico, 2020). In the Ghanaian context, Gen Xers were raised in a period where the country's political environment witnessed frequent coups d'état (Adjaye, 2023). They prioritize achieving a balance between work and personal life than earlier generations, ensuring they devote ample time to their family (Mahmoud et al., 2021; Safi, 2019). Often referred to as the 'latchkey' cohort, they are recognized for their strong sense of autonomy and inquiring attitude (Nelson & Duxbury, 2021; Perkins & Herring, 2021). Though members of this cohort are often tagged as digital immigrants, they exhibit adaptability, technological proficiency, and analytical reasoning skills (T. Brown, 2011). Gen Xers effectively serve as a connection between the analog and digital worlds.

2.6.2 Gen Y

Gen Y, also known as Millennials, were born between 1982 and 1999. They are considered the first generation to grow up in the millennium (Mahmoud et al., 2021). This cohort is characterized by their comfort with technology as they were raised during a time of internet and mobile device advancements (Johnson et al., 2016; Perkins & Herring, 2021). Millennials are commonly perceived as placing importance on finding purpose and significance in their careers, prioritizing flexibility in their work arrangements over strict hierarchies (Safi, 2019). The oldest members of this cohort experienced the 1983 drought and hunger, albeit when they were babies (Adjaye, 2023). These events might have shaped the Ghanaian Gen Yers, making them more resilient and adaptable.

2.6.3 Gen Z

Generation Z, born between 2000 and 2012, are considered the true digital natives (Mahmoud et al., 2021). Gen Zers are the youngest entering the workforce and their upbringing has been greatly influenced by social media, and worldwide connectivity (Maan & Srivastava, 2023; Mahmoud et al., 2021). Considered as the most tech-savvy at the workplace currently, they are much familiar with the use of social media applications such as Facebook, Twitter (X), Whatsapp, and Skype among others for communication (Kolog et al., 2024; Maan & Srivastava, 2023). In the Ghanaian context, this cohort was born during the fourth republic, when the country embraced democratic governance with elections consistently used to replace governments (Adjaye, 2023). Therefore, this cohort is likely to prefer work environments where personal views are respected.

2.6.4 Age Bias and Employment

Age biases about employment in the tech industry are a concern in today's workplaces (Stypińska et al., 2023). Ageism, which involves stereotypes and bias based on age, is a prevalent issue in the tech industry (Hoover, 2024). A survey conducted by Dice (2018), found that 76% of tech workers believe ageism is prevalent in the tech industry worldwide. Also, 80% of tech workers in their late 40s expressed concerns about how their age could impact their career progression. Furthermore, according to Gullette (2017), Silicon Valley, the US enclave for cutting-edge technology, may perhaps be the most ageist place in the world. Unfortunately, statistics on the median age of workers at Big Tech companies do not offer any relief. According to Payscale (2021), the median age of workers at Facebook stands at 28, Google at 30, LinkedIn and SalesForce at 29, and Microsoft at 33.

Although a plethora of anecdotal evidence exists concerning ageism in the tech industry, the reasons are often not straightforward (Stypińska et al., 2023). The rapidly and ever-evolving landscape of the technology sector often values young people as they are considered to be creative and easily adaptable to emerging technologies (Lyons, 2016). As illustrated by a blunt statement from Facebook CEO Mark Zuckerberg, "Young people are just smarter" (Johnston, 2016). Regrettably, age biases are not limited to hiring decisions. It manifests in varied forms within an organization such as the organizational culture, the kinds of opportunities for professional development offered, and the perceptions about older employees' capacity to drive innovation (Hoover, 2024; Lyons, 2016).

2.7 Programming Languages and Employment

In the ever-evolving field of software engineering, proficiency in programming languages is key to unlocking job opportunities and advancing one's career (Xie et al., 2019). This section delves into the significance of knowledge in programming languages and its influence on employment in the software engineering industry.

2.7.1 Overview of Programming Languages

A programming language consists of instructions that guide the computer in carrying out tasks. These instructions are often seen as code organized according to the syntax of the programming language (Gabbrielli & Martini, 2023; Xie et al., 2019). Programming languages form the foundation of software development, equipping developers with the requisite tools to develop applications, websites, and other software solutions (Lee, 2017). Ranging from low-level languages such as 'C' and assembly language to high-level languages like Python and Java, each programming language has its unique syntax, functionalities, and applications (Gabbrielli & Martini, 2023). Software engineers need to understand the features and capabilities of

different programming languages to effectively convey commands and develop functional software solutions (Xie et al., 2019).

2.7.2 Importance of Programming Language Knowledge in Software Engineering

Proficiency in programming languages is highly desired in the software engineering field for several reasons. To start with, being knowledgeable in programming languages enables software engineers to write code, troubleshoot software issues, and maintain developed software (Gabbrielli & Martini, 2023). Moreover, being familiar with programming languages broadens the scope of projects and job prospects for software engineers since different languages are preferred for different applications and industries (Lee, 2017). For instance, whereas languages such as JavaScript, Python, Ruby, and PHP are required for Website development, React Native, Dart, Swift, among others are required for Mobile development. Furthermore keeping up to date with programming languages and technologies showcases adaptability and a dedication to growth, qualities that are appealing to employers (Xie et al., 2019).

2.7.3 Programming Language Knowledge and Employment

The level of proficiency in programming languages plays a role in determining the employment prospects and career growth of software engineers (Ibezim & Chibuogwu, 2017). Job listings frequently outline the programming languages that are either required or preferred and candidates showcasing expertise in these languages stand a chance of landing job offers (Jebreen & Nabot, 2021; Stamm, 2023). Furthermore possessing expertise in niche programming languages such as Kotlin, can set candidates apart in competitive job markets, potentially resulting in increased salaries and opportunities for career growth (Camus, 2021).

Conversely, a lack of proficiency in programming languages could restrict job options and impede career growth within the SE industry (Tirado, 2021).

2.8 Software Engineering Competencies

In 1953, David McClelland, an American management theorist, discovered a quality in humans that he referred to as "competence" (Chouhan & Srivastava, 2014). The concept of competencies gained popularity in 1973 when McClelland introduced and examined it in a research paper titled "Testing for Competence Rather than Intelligence" (McClelland, 1973). Since then competencies have been acknowledged as indicators of employee performance, alongside an individual's knowledge and skills as measured by test results (Kansal et al., 2012).

Competencies are understood differently by scholars. According to Chouhan and Srivastava (2014), "Competencies include the collection of success factors necessary for achieving important results in a specific job or work role in a particular organization." Wagenaar (2014) refers to competencies as "a dynamic representation of demonstrated knowledge, understanding/ insight/ comprehension, (subject specific and generic) intellectual, practical and interpersonal skills and (ethical) values". The meaning of competencies varies depending on the context, as each environment defines its unique combination of knowledge and skills that illustrate competence (Voorhees, 2001). Hence, in the context of this study, 'competencies' refers to the combination of knowledge, skills, and abilities required by software professionals for effective performance in their assigned tasks.

2.9 Empirical Review of Existing Literature on SEC

This section provides a review of existing SEC literature derived from academic literature. This will provide useful insights into already covered SEC research domains, which will help in identifying research gaps that will inform this study. Table 2.2 provides a summary of existing SEC literature selected for the study.

Table 2.2 Summary of SEC Literature

Author(s)	Purpose of the study	Context and methodology	Theories/SE Framework/Model Used	Results/Findings
Stamm (2023)	To identify the most sought-after attributes in new hires	Mixed method (Interview and survey); USA	Classification approach based on Attributes, Professional Skills, and Technical Areas.	Identified 42 attributes sought by employers in new hires, with professional skills such as teamwork ranking very high.
Akdur (2022)	To identify and analyze the gap between the skills taught in academia and those required in the software engineering industry.	Quantitative (survey of 628 software practitioners); Turkey	N/A	Identified Software Configuration Management as the most significant gap between the skills taught in universities and those required in the industry. The study also found critical thinking, teamwork and communication as the most important soft skills
Assyne et al. (2022)	To develop a holistic framework for determining the essential competencies for software development that can be customized according to the characteristics of a particular software project.	Qualitative; Interviews with 138 industry professionals in Norway	Kano Model and the Competency Framework for Software Engineers	Proposed the Unified Competence Gate for Software Professionals (UComGSP); Identified 63 soft and 62 hard competencies; 25 essential competencies

Galster et al. (2022)	To investigate the relevant soft skills in the New Zealand software industry	Qualitative; analyzed job postings from online job portal in New Zealand	N/A	Identified 17 soft skills with communication-related soft skills found to be most in demand
Rabelo et al. (2022)	To determine the non-technical skills demanded in the SE market	Mixed method; analyzing job postings, statistical analysis, interviews; Europe: Germany, Netherlands, and the United Kingdom.	N/A	Identified 30 non-technical skills that are demanded by employers, with 'Teamwork' leading the list.
Jebreen and Nabot (2021)	To investigate the SE skills demanded by software firms in Jordan	Quantitative; analysis of job advertisements; Jordan	Classification approach based on Technical and Soft skills	Identified 6 technical and 9 soft skills that are in-demand by employers in Jordan
Ntinda et al. (2021)	To ascertain the skills required of SE graduates by the local software industry	Qualitative; Focus group method involving SE professionals; Namibia	N/A	Identification of 3 technical and 9 non-technical skills required of SE graduates by the local software industry
Marebane and Hans (2021)	The study investigates how well undergraduate computing programs at South African Universities of Technology (UoTs) incorporate software engineering ethics into their curricula.	Qualitative (content analysis of curriculum documents); South Africa	N/A	The study revealed insufficient coverage of software engineering ethics in the curricula of UoTs in South Africa, providing only limited opportunities for students to develop necessary ethical competencies.

Groeneveld et al. (2020)	To identify and rank non-cognitive abilities needed for a developer to excel in the SE industry.	Qualitative; Delphi approach, inviting 36 experts from 11 different countries, with a concentration from Belgium	Through an inductive approach employed classification of non-cognitive competencies based on communicative, collaborative, problem-solving, and personal skills	Identified and ranked a list of 55 non-cognitive skills classified into four major areas: communicative skills, collaborative skills, problem-solving skills, and personal skills
Mangiza and Brown (2020)	To identify requisite skills for software developers in startups	Qualitative (Delphi approach); Interviews with expert software developers in startups; South Africa	The SDLC model	Identified 43 unique skills, both technical and soft; top 19 skills were predominantly soft skills
Oguz and Oguz (2019)	To explore and understand the persistent gap between the demands of the software industry and the curricula of software engineering education.	Mixed-method approach (questionnaires, interviews and analysis of job ads); Turkey	N/A	The study found that academic software training doesn't fully align with industry expectations. To bridge this gap, curricula should prioritize practical, large-scale projects that reflect industry standards.
Hiranrat and Harncharnchai (2018)	To identify essential technical and soft skills for software development positions in Thailand	Quantitative (Text mining from online job portals); Descriptive statistical method for analysis; Thailand	Soft skills, explicit technical skills and implicit technical skills	Identified 20 instances each of frequently occurring soft skills, explicit technical skills, and implicit technical skills

Kusumasari et al. (2018)	To present a list of competencies that are important in a software development team	Quantitative (literature review and questionnaire); Indonesia	Classification of competencies based on hard and soft competencies	Identified 57 competency elements out of which 24 were ranked as a top priority
Wang et al. (2018)	To determine the qualifications and experience Canadian employers seek in requirements engineers.	Qualitative (analysis of job ads); Canada	N/A	The most sought-after requirements engineering (RE) skills were RE methods and project management skills.
Calazans et al. (2017)	To determine the competencies expected by the Brazilian and Mexican markets for software requirements professionals	Qualitative (content analysis of job ads); Brazil and Mexico	KSA competency model (knowledge, skills and abilities)	The study revealed that higher education is crucial in both countries, with key skills including methodological competence and communication, and important attitudes being analytical thinking, organization, and interpersonal relationships.
Jia et al. (2017)	To understand the soft skills requirements for mobile applications developers	Qualitative; Analysis of online job postings in China using text-mining techniques including word segmentation, similarity calculation, and clustering.	N/A	Identified 13 categories of soft skills with communication' and teamwork ranked as the most important soft skills

2.9.1 Analysis of the Summarized SEC Literature

The reviewed literature suggests that SEC studies have been carried out in diverse geographical contexts. The studies encompass a range of locations spanning from Europe (Norway, Belgium, the Netherlands, the UK, and Germany), North America (the USA), Asia (China, Thailand, and Indonesia) to Africa (Namibia and South Africa) and Latin America (Brazil and Mexico). This underscores the importance of SEC from a global perspective.

In addition, diverse methodological approaches have been employed by the reviewed studies to explore and investigate SEC. Several studies (Assyne et al., 2022; Galster et al., 2022; Groeneveld et al., 2020; Ntinda et al., 2021) employed qualitative methodologies such as interviews, case studies, and focus group discussions. This allowed for an in-depth and nuanced understanding of the essential competencies required of software professionals. Additionally, some of the studies relied on secondary data, which involved analyzing SE job postings to identify the competencies required by the industry (Calazans et al., 2017; Jia et al., 2017; Wang et al., 2018). For instance, Jia et al. (2017), used text-mining techniques such as word segmentation, similarity calculation, and clustering to identify and analyze online SE job postings in China. Moreover, the use of quantitative methodologies such as surveys by Kusumasari et al. (2018) provided statistical perspectives on SEC. Furthermore, the mixed methods approach used by Stamm (2023), offered a holistic perspective, allowing for both depth and breadth in the findings.

The empirical findings of the reviewed literature reveal several competencies required by the software industry. These competencies are usually grouped into two broad classifications: technical (hard) and soft competencies. Whereas several studies explored both technical and soft competencies required by the software industry, others concentrate primarily on soft skills.

As summarized in Table 2.2, several studies propose frameworks or models that explicitly incorporate soft skills as a crucial component. For example, the study by Assyne et al. (2022), reported more soft competencies (63) than hard competencies (62). Similarly, Mangiza and Brown (2020) discovered that four out of the top five skills identified were soft skills, which highlights the importance of attributes like problem-solving and teamwork in startup environments. Additionally, several studies (Galster et al., 2022; Groeneveld et al., 2020; Jia et al., 2017; Rabelo et al., 2022) are exclusively dedicated to investigating soft skills, which indicates a growing acknowledgment in the research community about the significance of soft skills in the SEC domain. Skills such as communication, teamwork, problem-solving, and adaptability are now considered very important in achieving success in SE (Jia et al., 2017; Rabelo et al., 2022). This represents a shift from the traditional perspective on SE, which focused heavily on technical abilities such as proficiency in programming languages, algorithms, and system design (Sedelmaier & Landes, 2015). This change indicates an approach to SE education and practice that aligns with the diverse demands of the industry.

Another discovery from the studies summarized in Table 2.2 is the consistent disparity between academic preparation and industry demands within the SE field. Multiple studies highlight a mismatch between the skills taught in educational settings and those required by the industry (Galster et al., 2022; Ntinda et al., 2021; Stamm, 2023). For instance, Ntinda et al. (2021) underscore the importance of both technical and non-technical skills in Namibia's local software industry suggesting that academic programs may not adequately equip students for real-world challenges. Similarly, Calazans et al. (2017) report discrepancies between what the academic research predicts and the expectations of the SE industry. The increasing significance of soft skills as discussed in the previous section further contributes to widening the gap between academia and industry. Educational programs often prioritize technical skills while

employers, particularly those within startup environments as noted by Mangiza & Brown (2020), are placing more value on soft skills such as problem-solving and teamwork.

To bridge the Industry-Academia gap, Ntinda et al. (2021) emphasize the need for collaboration between institutions and industry to ensure that graduates are well-equipped for industry. This viewpoint finds support albeit indirect from authors such as Assyne et al. (2022) who propose a comprehensive framework, the UComGSP, for better alignment of competencies between educational institutions and employers. In conclusion, bridging the gap between SE competencies in academia and industry is multifaceted, encompassing skill mismatches and varying regional expectations. Addressing this challenge will require collaborative efforts from both educational institutions and industry players.

2.9.2 Research Gaps and Directions for Future Research

The reviewed literature reveals notable gaps that are worth considering for future research. First, several existing SEC literature have sought to understand the competencies that are valued by employers (Jebreen & Nabot, 2021; Mangiza & Brown, 2020), whereas others have also sought to investigate the skill gap between what academia teaches and what industry wants (Galster et al., 2022; Ntinda et al., 2021). However, to the best knowledge of the researcher, no existing study has explored how the various competencies influence employment outcomes. Also, although there has been anecdotal evidence suggesting a prevalence of generational age biases concerning employment in the SE industry, empirical evidence regarding the phenomenon is scarce (Stypińska et al., 2023). Moreover, the matter of knowledge in programming languages and it moderating influence on employment outcomes have not been covered by existing literature. As such these issues can be explored by future research.

In addition, while several research studies have been conducted in the developed country context (Assyne et al., 2022; Calazans et al., 2017), there is a dearth of SEC studies in both the Ghanaian context and the broader African context. Moreover, as reported by Calazans et al. (2017), the competencies required of software engineers can be significantly influenced by geographical differences. Hence the need to explore SEC in the Ghanaian and the African context.

Furthermore, there is a gap in the use of theory to understand how competencies influence employment. Existing literature have relied on theories such as the Theory of Planned Behavior, and the Social Cognitive Career Theory to understand how individuals decide their career paths (Gregory & Penela, 2023; Tam et al., 2024). However, these theories do not explain why individuals develop certain competencies. Hence the use of other relevant theories is needed to explore the phenomenon under study.

2.10 Chapter Summary

This chapter explored vital concepts in SE while reviewing existing SEC literature. In addition, the chapter offered a comprehensive overview of the SE landscape in Ghana, including a look at the industry's prospects as well as related government policy interventions. Moreover, other vital concepts such as generational age and programming languages were explored. Furthermore, an empirical review of existing SEC literature was conducted to reveal what has been covered by existing literature. All of the above contributed to identifying gaps in existing literature that will be addressed in subsequent chapters of this study.

CHAPTER THREE

THEORETICAL FRAMEWORK

3.1 Chapter Overview

In the previous chapter, an in-depth review of existing literature was undertaken to understand the fundamental aspects of SE and the unique environment of SE in Ghana. This included examining government policies and initiatives that influence the field. This extensive literature review laid the groundwork for comprehending the competencies needed in the SE field. This chapter presents the theoretical framework of the study. To achieve this goal, this chapter reviews theories that are applied in employment studies such as the Theory of Planned Behavior, Social Cognitive Career Theory, and the Human Capital theory. This is followed by the justification for the selection of Human Capital Theory as well as the conceptual framework used for the study.

3.2 Employment Theoretical Models

The reasons why and how persons gain employment is a major area of focus for researchers. Over time, researchers have strived to understand, predict, and explain the factors that influence the employment of individuals in various industries. While the use of theories on employability and employment is scarce (Forrier et al., 2018), researchers have explored various theories and models to understand how individuals enhance their employability to secure employment and career progression (Gregory & Penela, 2023; Tam et al., 2024). These include the Theory of Planned Behavior (TPB), Social Cognitive Career Theory (SCCT), and the Human Capital Theory (HCT).

3.2.1 Theory of Planned Behavior

The Theory of Planned Behavior (TPB), developed by Ajzen (1991), is a widely used theoretical framework in psychology and behavioral science research. According to TPB, a person's decision to engage in an action or behavior is primarily driven by their behavioral intention to do so. The theory extends the theory of reasoned action (TRA) by incorporating the concept of perceived behavioral control to account for behaviors that are beyond one's control (Kan & Fabrigar, 2017). TPB posits that an intention to engage in a particular behavior is shaped by three factors, constituting the major constructs of the theory (Ajzen, 1991):

Attitude toward the Behavior: A person's opinion of a behavior, whether positive or negative, influences the likelihood of performing the behavior. A positive attitude is linked to a chance of engaging in the behavior whereas a negative attitude about the behavior will refrain them from doing so (Ajzen, 1991; Kan & Fabrigar, 2017).

Subjective Norms: Social expectations play a role in influencing one's decision to either engage in or refrain from behaviors. An individual's perception of how people they value or consider important think of them performing a behavior is a key determinant of whether they will engage in such behavior. Thus subjective norms typically involve evaluating how the approval of important people in one's life gives to the individual carrying out the behavior (Ajzen, 1991; Kan & Fabrigar, 2017).

Perceived Behavioral Control: This factor signifies how easy or hard someone perceives it to carry out the behavior in question. It includes factors like self-efficacy (belief in one's capability to perform the behavior) and the perceived level of control over the behavior. Hence,

when an individual perceives enormous challenges or a lack of skill, their motivation to perform the behavior in question weakens (Ajzen, 1991).

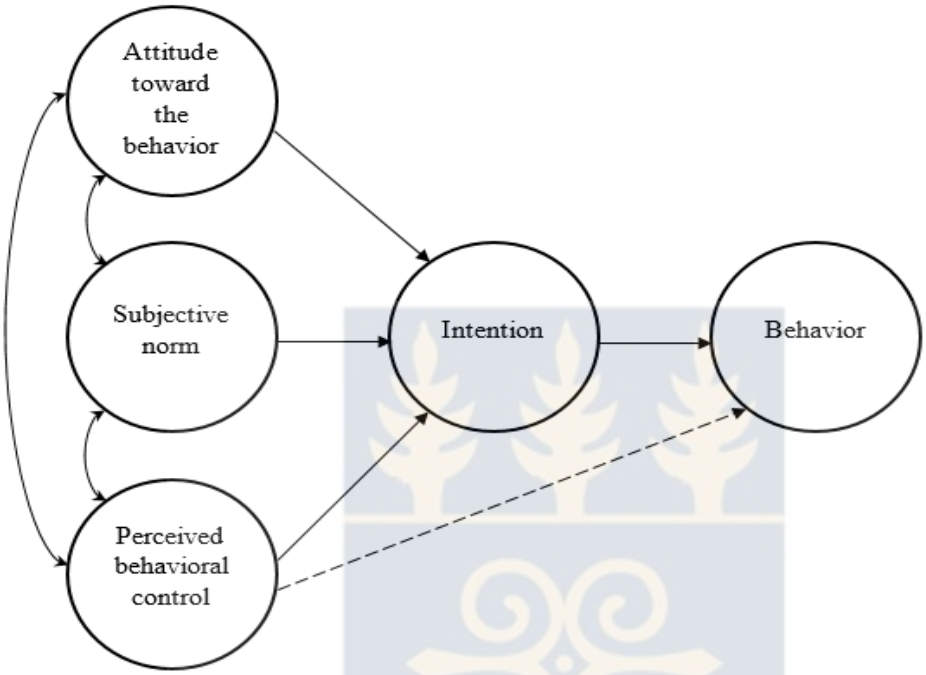

Figure 3.1 Theory of Planned Behavior

Source: Ajzen (1991)

3.2.1.1 Weaknesses of TPB

A number of weaknesses of TPB have been identified. Firstly, the theory assumes that individuals have the opportunities and resources to succeed in carrying out the behavior they desire. Thus, the theory overlooks economic variables that could affect an individual's decision to engage in a behavior. Secondly, disagreements exist on the interchangeability of perceived behavioral control and self-efficacy (Armitage & Conner, 2001; Johnson & Hall, 2005).

3.2.2 Social Cognitive Career Theory

The Social Cognitive Career Theory (SCCT) advanced by Lent et al. (1994), is a theoretical framework that helps in understanding how individuals develop their career and academic interests, the decisions made regarding those pursuits, and the performance achievements that result from those decisions. SCCT is an extension of Bandura's (1986) Social Cognitive Theory by emphasizing how people navigate their career paths through personal initiative such as goal setting while taking into account the influence of external factors such as educational opportunities. The key constructs of SCCT are explained below:

Self-Efficacy: This refers to one's confidence in their ability to perform a given set of tasks and overcome obstacles associated with a career path. Thus, individuals are more likely to choose a particular career path when they have high self-efficacy in undertaking the tasks associated with it.

Outcome Expectations: These are the expected outcomes of pursuing a chosen career path. This means individuals are more likely to choose a particular career path when they expect favorable outcomes such as financial stability, fulfillment, or satisfaction.

Interest: SCCT establishes that individuals develop enduring interests in activities where they have self-efficacy and expect positive outcomes. Thus, individuals are more likely to choose a particular career path when they have an enduring interest in that career due to their confidence in performing the tasks associated with it and the favorable outcomes expected.

Goals Setting: SCCT lay emphasis on the importance of establishing ambitious and achievable goals in advancing one's career. The theory posits that goals offer guidance and motivation,

hence individuals with clearly defined goals tend to make strategic career decisions which leads to performance attainments such as fulfillment and skill development.

Contextual factors: These are the external factors that influences an individual's choice of a career path. This includes, supportive family, mentors, educational opportunities, and other societal influences.

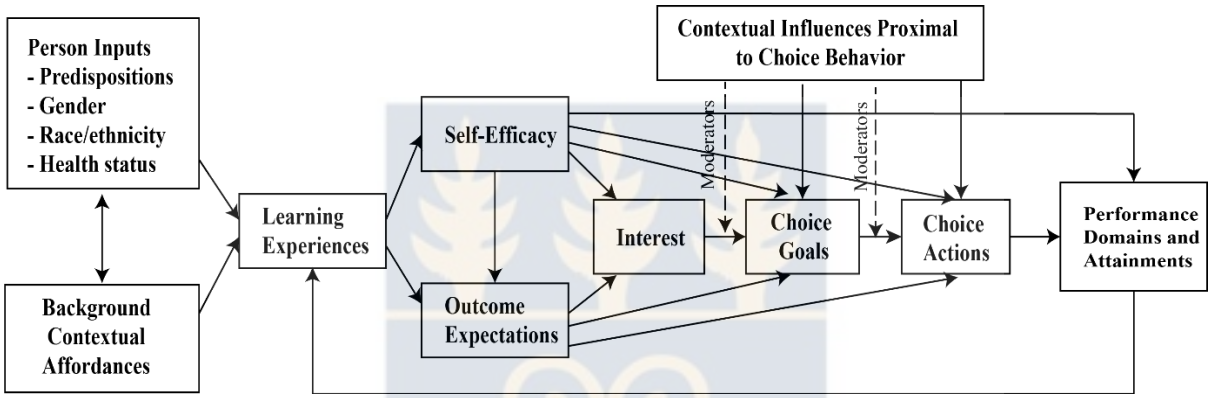

Figure 3.2 Social Cognitive Career Theory (SCCT)

Source: Adopted from Lent et al. (1994)

3.2.3 Human Capital Theory

The Human Capital Theory (HCT), first introduced by economists Schultz (1961) and Becker (1962), considers individuals as investors in their own skills, knowledge, and abilities (Nerdrum & Erikson, 2001). According to this theory, investments in oneself, such as education and training, are akin to investing in assets. Such a decision to build one's human capital can enhance job prospects while increasing productivity and earning capacity (Law, 2010). Scholars have widely debated and interpreted the human capital theory in various ways. Whereas some scholars like Daeboug (2009), contend that knowledge and skills obtained via education constitute human capital, others, such as Blundell et al. (1999), argue that human capital is acquired through a combination of innate abilities and skills acquisition through formal education and training.

Although HCT is widely utilized in existing research, there exist challenges in measuring human capital due to the use of inconsistent constructs (Mubarik et al., 2017). However, the core concept behind these constructs consistently refers to the accumulation of competencies gained through formal and informal education, training, innate abilities, and relevant work experience. For instance, Marimuthu et al. (2009) identified training, education, knowledge, and skills as the main constructs of human capital. Thus, the authors consider training, education, knowledge, and skills acquisition as important elements that constitute human capital, which eventually translate into both financial and non-financial performance for organizations. This is depicted in Figure 3.3.

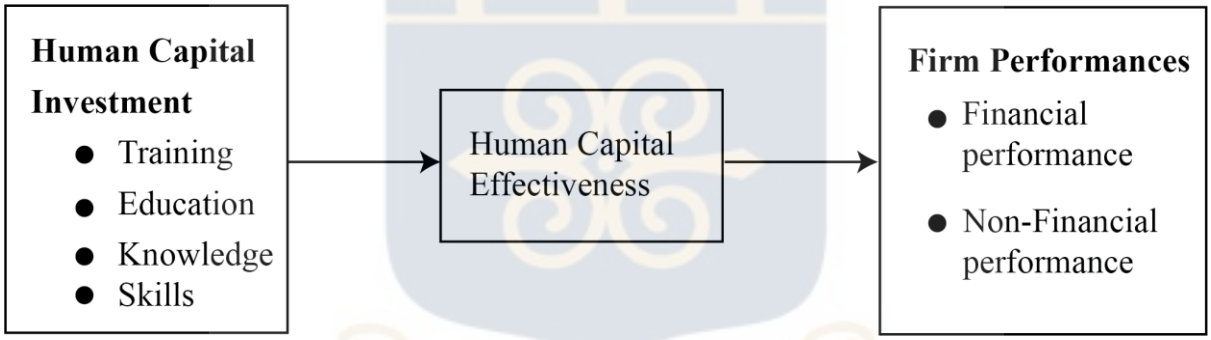

Figure 3.3 HCT Framework

Source: Adopted from Marimuthu et al. (2009)

3.2.3.1 Justification for Choosing the Human Capital Theory (HCT)

To understand why individuals secure employment in SE, it is important to apply a theory that explains the intricate factors that influence employment. Given the heavy reliance of the SE industry on its professionals' competencies (Assyne et al., 2022), HCT offers a way to understand how investment in software engineering skills can be considered valuable assets by software companies. HCT posits that investments in human capital, such as education, training, certifications, and hands-on experience, increase an individual's value in the labor market. In

the context of SE, this translates to software professionals cultivating sought-after competencies that enhance their job prospects (Li et al., 2020). This theory therefore aligns with the objectives of the study as it supports the exploration of how investments in software engineering competencies translate into job opportunities and career progression.

In addition, HCT has been employed in numerous studies across different disciplines and countries, linking investment in competencies or skills development to employment and organizational performance. For example, HCT has been applied in research conducted in countries such as Ghana (Adom & Asare-Yeboa, 2016), China (Law, 2010), and the United States (Cornacchione & Daugherty, 2013). It has also been employed in diverse fields, including entrepreneurship (Mubarik et al., 2017), economics (Marimuthu et al., 2009), software engineering (Bontis et al., 2007), and employability studies (Suleman, 2021). This versatility highlights HCT's effectiveness in understanding how investments in human capital influence various outcomes such as career progression and firm performance.

In applying HCT to the field of Software Engineering (SE), it is crucial to consider the specific skillsets and knowledge valued in the industry. While Becker (1962) identified core constructs like education, training, and healthcare, researchers often adapt these constructs to suit specific contexts. For instance, while Kabia (2011) considers formal education, firm specific on-the-job training, and other knowledge/vocational qualifications as key constructs of human capital, Adom and Asare-Yeboa (2016) focus on 'level of education, area of education, business training and experience from previous employment' as the main constructs of women entrepreneur's human capital.

In the context of this study, the key constructs of human capital are derived from the main software engineering competency domains proposed by Liu et al. (2016). The authors (Liu et al., 2016) categorize SEC into six major domains: achievement orientation, level of knowledge, communication skills, thinking skills, service orientation, and teamwork. This study however adapts 'level of knowledge' to 'domain knowledge', 'thinking skills' to 'problem solving ability' and 'service orientation' to 'professional work ethic' for better alignment with existing literature. Liu et al. (2016) refer to 'level of knowledge' as the level of professional knowledge, a description that is akin to 'domain knowledge' – a term widely used in SE to denote the requisite technical and business knowledge required for SE roles (Niknafs & Berry, 2017). This adaptation is further justified due to the use of the term 'domain knowledge' in several SEC literature (Jebreen & Nabot, 2021; Ntinda et al., 2021; Zhang et al., 2011) and job postings. Moreover, Liu et al. (2016) refer to 'thinking skills' as including the ability to analyze and solve problems, but in several SEC studies (Assyne et al., 2022; Hiranrat & Harncharnchai, 2018; Ntinda et al., 2021), such description is usually denoted as 'problem-solving ability'. Furthermore, Liu et al. (2016) defines 'service orientation' as including integrity, self-discipline, and a task-oriented attitude, attributes that are akin to professional work ethic ability (Mangiza & Brown, 2020).

In addition, two other variables, 'knowledge in programming languages' and 'generational age', are deployed in this study as moderators. Although the use of moderators are lacking in SEC studies, many forms of moderators have been used in related employability studies. These include gender (Awodiji, 2024; Hinai et al., 2020; Sapkota et al., 2024), self-efficacy (Razzak et al., 2023; Waziri et al., 2024), culture values (Moy et al., 2023), career attitude (Presti et al., 2024), and generational age (Moreira et al., 2018; Otterbach & Sousa-Poza, 2016). For instance, in past studies such as Moreira et al. (2018), generational age was deployed as a

moderating variable in examining the relationship between organizational competencies development practices and internal employability. Similarly, in an M-banking usage study, Çera et al. (2020) found that generational age (Gen Y and Z) significantly moderates the relationship between gamification and behavioural intention. In software engineering, anecdotal evidence suggests that age is often considered an important factor during recruitment (Stypińska et al., 2023), likely due to perceptions regarding the tech-savviness and innovativeness associated with certain age groups. Hence, this study builds on related research, such as that of Moreira et al. (2018), to further explore the construct of generational age within the context of the SEC domain. Regarding knowledge in programming languages, although its use as a moderating variable is lacking, past studies such as Tirado (2021), highlight how proficiency in this competency enables software professionals to solve problems more effectively, justifying its inclusion in this study. Thus, this study contends that the main SEC domains represent investments in human capital, which aid software professionals in attaining employment, moderated by their knowledge in programming languages and their generational age. The study, therefore, modifies and extends HCT to better understand the underlying factors that help software professionals gain employment.

3.3 Development of the Conceptual Framework

3.3.1 Achievement Orientation and Employment

Achievement orientation reflects a software engineer's inherent drive to consistently establish and meet high-performance goals through persistence, self-discipline, and a strong desire for continuous improvement (Liu et al., 2016; Tenzer & Yang, 2020). This multifaceted competency includes an individual's drive for excellence, tenacity, self-motivation, confidence in oneself, and self-evaluation (Kao et al., 2022; Liu et al., 2016). In SE, technology evolves quickly and there is a need for innovative solutions (Calazans et al., 2017). Possessing these

qualities can greatly influence one's job prospects and career progression. Tenacity highlights an individual's ability to persevere when confronted with challenging and difficult situations. Software engineering involves a lot of problem solving which makes tenacity a much desirable trait of software engineers. Tenacity allows software engineers to navigate the myriad of challenges they may encounter in developing and debugging applications (Vadlamani & Baysal, 2020). Self-motivation is very essential as the SE field requires continuous learning and adaption. When software engineers are self-motivated, they can stay up to date with programming languages, emerging technologies and trends without any external stimuli (Daneva et al., 2017). This makes software engineers valuable team members for any software project. Self-confidence allows software engineers to have faith in their abilities and decisions, take calculated risks, and advocate their ideas resulting in innovative solutions that advance the industry (Galster et al., 2022). Employers usually look for individuals who demonstrate belief in their skills in terms of leading teams and managing projects. Self-positioning and evaluation require assessing one's skills, acknowledging areas for growth, and strategically placing oneself in roles or projects where one can make a significant impact are essential for personal development (Daneva et al., 2017). It ensures that software engineers are utilized in positions where they can add value to their team and the organization (Vadlamani & Baysal, 2020). Given the arguments raised, the study proposes that:

Proposition 1: Achievement orientation, characterized by tenacity, self-motivation, self-confidence, self-positioning, and self-evaluation, influences employment outcomes of software engineers.

3.3.2 Domain Knowledge and Employment

Domain knowledge refers to extensive technical and business knowledge in a specialized SE field (Liu et al., 2016; Niknafs & Berry, 2017). Software engineering has several in a particular

SE domain. domains that cover the diverse specializations within the field. These domains include website development, mobile application development, data science, machine learning, game development, and cloud computing among others (Aggarwal, 2023). Software engineers are usually employed in any of these domains and are therefore required to possess the requisite technical expertise that is needed in the respective domain (Jebreen & Nabot, 2021). Possessing domain knowledge of the specialized field sets a software engineer apart thereby enhancing their employability (Niknafs & Berry, 2017). Domain knowledge in SE goes beyond having technical know-how, it also involves the ability to apply that knowledge to solve problems and come up with ideas to drive innovation in the industry. It requires continuous learning, staying updated with the relevant knowledge and tools, and applying them to enhance professional practice (Aggarwal, 2023). Given the points raised, the study proposes that:

Proposition 2: Domain knowledge have an influence on the employment outcomes of software engineers.

3.3.3 Communication Skills and Employment

Communication skills refer to the ability to convey clear and timely information, provide prompt feedback, and translate technical concepts into more user-friendly language (Ahmed et al., 2013; Liu et al., 2016). Software engineering rarely occurs in isolation. Software engineers usually work with team members, clients, and other relevant stakeholders (Oguz & Oguz, 2019). Hence the software engineer should possess strong communication skills to effectively collaborate with team members and relevant stakeholders (Semerikov et al., 2020). This involves building strong working relationships, providing and receiving feedback, and resolving any conflicts. Communication skills is crucial in many facets of software engineering (Jia et al., 2017). For instance, at the requirements gathering stage, communication skills enable software engineers to ask the right questions, actively listen, and translate technical concepts

in a way that non-technical users can understand (Calazans et al., 2017). By focusing on end-users, software projects that meet their user expectations are provided (Semerikov et al., 2020). Moreover, software engineers who can share knowledge and mentor less experienced colleagues create a positive learning environment thereby contributing to the overall growth and development of their team members (Oguz & Oguz, 2019). Also, strong written communication skills allow code project requirements, architecture, and design choices understandable to developers, making it easier to maintain and update the software (Akdur, 2021). Furthermore, in some cases, software engineers may have to present project plans, research findings, or technical proposals to stakeholders. Being able to communicate and present information effectively is vital, in such scenarios to ensure clarity, secure support, and influence decisions. Given the arguments raised, the study therefore proposes that:

Proposition 3: Communication skills has an influence on the employment outcomes of software engineers.

3.3.4 Problem-solving Ability and Employment

Problem-solving ability refers to the ability of a software engineer to think critically, analyze complex challenges, and devise innovative solutions to those challenges (Schefer-Wenzl & Miladinovic, 2019). In reality, most software projects do not unfold exactly as planned. Software engineers are usually faced with complex problems and challenges daily, ranging from simple syntax errors to complex algorithm design (Schefer-Wenzl & Miladinovic, 2019). Hence the ability to think critically and develop alternative strategies when faced with challenges is crucial for software engineers. Having problem-solving abilities is key to mitigating setbacks while keeping projects running smoothly (Oguz & Oguz, 2019). Problem-solving goes beyond repairing what is not working. Skilled problem solvers can think creatively, explore unconventional solutions, and come up with ideas that enhance current

systems or create entirely new functionalities (Özyurt, 2015). This ability distinguishes them as contributors who drive innovations. Effective problem-solving requires examining information, evaluating potential solutions, and making decisions that advance project goals (Oguz & Oguz, 2019). Software engineers have the ability to troubleshoot and overcome obstacles independently, making them valuable team members who can skillfully navigate challenges (Özyurt, 2015). Moreover, problem-solving abilities are essential in leadership positions, where strategic problem-solving and decision making play a role in overcoming business obstacles and driving organizational success (Mumford et al., 2017). Given the arguments raised, the study therefore proposes that:

Proposition 4: Problem-solving ability influences the employment outcomes of software engineers.

3.3.5 Professional Work Ethic Ability and Employment

Professional work ethic ability refers to the ability of a software engineer to adhere to high standards of integrity, self-discipline, and responsibility in their work (Sapada et al., 2017). Software engineering requires professional work ethics such as integrity, self-discipline, hard work, time management, task-oriented attitude, among others (Galster et al., 2022). In the software engineering industry, having a strong professional work ethic is highly valued. Those who exhibit integrity, honesty, discipline, ability to meet deadlines, a task-oriented attitude, and a willingness to work hard are more likely to land jobs, excel in their respective roles, and enjoy career growth (Stamm, 2023). Employers value individuals who take charge of their duties and display motivation (Semerikov et al., 2020). For instance, integrity helps in building trust and creates a positive work environment while discipline and focus help engineers manage their time effectively and produce quality results even under pressure. Employees with a strong

work ethic are able to work under pressure and meet stringent timelines, contributing to the firm's success (Kusumasari et al., 2018). Given the arguments raised, the study proposes that:

Proposition 5: Professional work ethic ability influences the employment outcomes of software engineers.

3.3.6 Teamwork Ability and Employment

Teamwork ability refers to a software engineer's ability to effectively collaborate with colleagues in a team environment, adapt to team members, and have a shared commitment to achieving common objectives (Lindsjørn et al., 2016; Liu et al., 2016). Software engineers usually do not work in isolation, they normally work in collaborative environments which makes teamwork an important skill set in any software engineering environment (Jia et al., 2017). Teamwork emphasizes effective communication, mutual respect, adaptability, and a collective focus on achieving common goals (Kusumasari et al., 2018). Thus, software engineers with teamwork ability are able to communicate effectively with other team members, respect colleagues, and adapt to the assigned team, while prioritizing the achievement of common goals (Semerikov et al., 2020). This is particularly important since the SE industry often executes complex software projects that require collaboration with other team members to leverage a diversity of skills, perspectives, and expertise to achieve project success (Stamm, 2023). Teamwork promotes a culture of learning and innovation where team members can exchange insights, question assumptions, and contribute to a culture of continuous improvement (Semerikov et al., 2020). Thus, software engineering is primarily about team effort, hence software engineers who can adapt their roles and working styles to accommodate team dynamics greatly enhance their employment outcomes and career progression (Weimar et al., 2017). Given the arguments raised, the study proposes that:

Proposition 6: Teamwork ability has an influence on the employment outcomes of software engineers.

3.3.7 Moderating Influence of Knowledge in Programming Languages

The study explores the moderating influence of Knowledge in Programming Languages on the relationship between some competency dimensions (Domain Knowledge and Problem-Solving Ability) and employment. By examining the moderating influence of knowledge in programming languages, this study aims to provide a more nuanced understanding of how these competency dimensions influence employment outcomes.

3.3.7.1 Moderating Influence of Knowledge in Programming Languages on Domain Knowledge and Employment

Having domain knowledge in a specific software engineering area is crucial as it enhances their employment outcomes. However, possessing knowledge in relevant programming languages plays a key role in how useful a software engineer's domain knowledge is when seeking jobs (Brown, 2019). This is because many programming languages have specialized tools, libraries, and frameworks that are tailored to development-specific tasks. Achieving mastery of these language-specific tools enhances the knowledge of software engineers, helping them achieve desired outcomes in the respective domain (Cheng, 2018). In addition, having extensive knowledge in programming languages that are highly sought after in a particular SE domain, can significantly enhance employability in that domain (Smuts & Smuts, 2022). For instance, within the Website development domain, a software engineer who possesses extensive knowledge in front-end languages such as JavaScript and its frameworks and knowledge in backend (server-side) languages such as Python, Ruby, and PHP, among others, will be highly sought after by employers (Brown, 2019). Given the arguments raised, the study proposes that:

Proposition 7: Knowledge in programming languages has a moderating influence on the relationship between domain knowledge and employment outcomes in software engineering.

3.3.7.2 Moderating Influence of Knowledge in Programming Languages on Problem-Solving Ability and Employment

In software engineering, the ability to solve complex problems is what distinguishes the average from the exceptional (Schefer-Wenzl & Miladinovic, 2019). However, the impact of a software professional's problem-solving abilities on job opportunities and career progression depends on their proficiency in programming languages (Smuts & Smuts, 2022). Being proficient in relevant programming languages enhances a software engineer's toolkit, enabling them to tackle problems from a broader perspective and with a diverse set of solutions (Ibezim & Chibuogwu, 2017). For employers, being proficient in programming languages is often interpreted as an indicator of a software engineer's ability to apply problem-solving skills in real-world scenarios (Tirado, 2021). This is because software projects come with their peculiarities, with different languages suited to different tasks. The ability to solve software-related problems therefore depends on how well one can utilize relevant programming languages (Abdulkareem & Abboud, 2021). Given the arguments raised, the study proposes that:

Proposition 8: Knowledge in programming languages has a moderating influence on the relationship between problem-solving ability and employment outcomes in the software engineering sector.

3.3.8 Moderating Influence of Generational Age

In addition to the moderating influence of Knowledge in Programming Languages, the study also explores the moderating impact of generational age on the relationship between

competency dimensions (Domain Knowledge, Communication Skills, and Professional Work Ethic Ability) and employment. This will shed light on potential differences in how Gen X, Gen Y, and Gen Z develop these competencies, ultimately affecting their employment outcomes.

3.3.8.1 Moderating Influence of Generational Age on Domain Knowledge and Employment

Generational age influences the link between the domain knowledge of software engineers and their job prospects in the software engineering industry. The generational cohort a person belongs to, be it Gen X, Gen Y, or Gen Z, can influence their behaviors and perceptions, and how they apply knowledge thereby affecting their career prospects (Maan & Srivastava, 2023). Thus, generational differences can affect how software engineers acquire domain knowledge, apply same within professional contexts, as well as how they adapt to emerging technologies (Fleming et al., 2017). Younger software engineers are perceived to bring new perspectives and are widely assumed to be more adaptable to new technologies and programming languages (Lyons, 2016). As a result, younger software engineers are more likely to be preferred in cutting-edge projects as they are deemed to be more familiar with the latest technological trends (Hoover, 2024).

On the other side, experienced software engineers from older generational cohorts may bring a wealth of expertise and a profound grasp of core concepts that come only with years of practice (Lyons, 2016). This extensive knowledge is vital in solving complex problems, mentoring less experienced and younger team members, and spearheading projects that demand familiarity with legacy systems as well as emerging technologies (Stypińska et al., 2023). Though organizations stand to benefit from having employees of different generational

cohorts, the perceptions about the various cohorts can affect the hiring decisions of software engineers. Given the arguments raised, the study proposes that:

Proposition 9: Generational age has a moderating influence on the relationship between domain knowledge and employment outcomes in the software engineering sector.

3.3.8.2 Moderating Influence of Generational Age on Communication Skills and Employment

Generational age plays a major role in the way communication skills affect employment outcomes in software engineering. This factor does not only influence how individuals communicate but also how their communication styles are perceived and valued in the workplace (Mahmoud et al., 2021). Existing literature suggests that different generations have diverse preferences regarding communication, which in turn affects collaboration, teamwork, and client interactions (Maan & Srivastava, 2023). Younger software engineers who are well-versed in digital communication tools and platforms tend to thrive in environments that prioritize quick, online collaboration (Talarico, 2020). Their familiarity with technology often results in the creation of communication strategies, which can be attractive, to employers who appreciate adaptability and tech-savviness. However, over-reliance on digital communication may not work in situations where, in-person interactions and traditional forms of communication hold significance (Fleming et al., 2017).

Older software engineers, on the contrary, may excel in face-to-face communication due to their upbringing in environments that value direct interactions (Maan & Srivastava, 2023). These interpersonal skills prove essential in scenarios involving negotiation, client management, and mentoring within teams. Their communication style often prioritizes clarity, depth, and the cultivation of long-term relationships, qualities that immensely contribute to

team cohesion and client satisfaction (Turner & Gurenlian, 2023). In light of the diverse perspectives and preferences regarding communication among the various generation cohorts, it can be proposed that:

Proposition 10: Generational age has a moderating influence on the relationship between communication skills and employment outcomes in the software engineering sector.

3.3.8.3 Moderating Influence of Generational Age on Problem-Solving Ability and Employment

Problem-solving has been widely recognized in existing literature as an important competency for software engineers (Mangiza & Brown, 2020; Ntinda et al., 2021). However, existing research suggests that the various generational cohorts may approach problem-solving differently, which could potentially impact their employment outcomes. Although the current literature has not extensively explored these differences, empirical evidence points to differences in the learning styles of the various generational cohorts (Deepa et al., 2021). These differences in learning may influence how individuals from different generations solve problems in the workplace. In light of this, this study seeks to explore how software engineers from various generational cohorts approach problem-solving and how these approaches might impact their employability, hence the proposition that:

Proposition 11: Generational age moderates the relationship between problem-solving ability and employment outcomes in the software engineering sector.

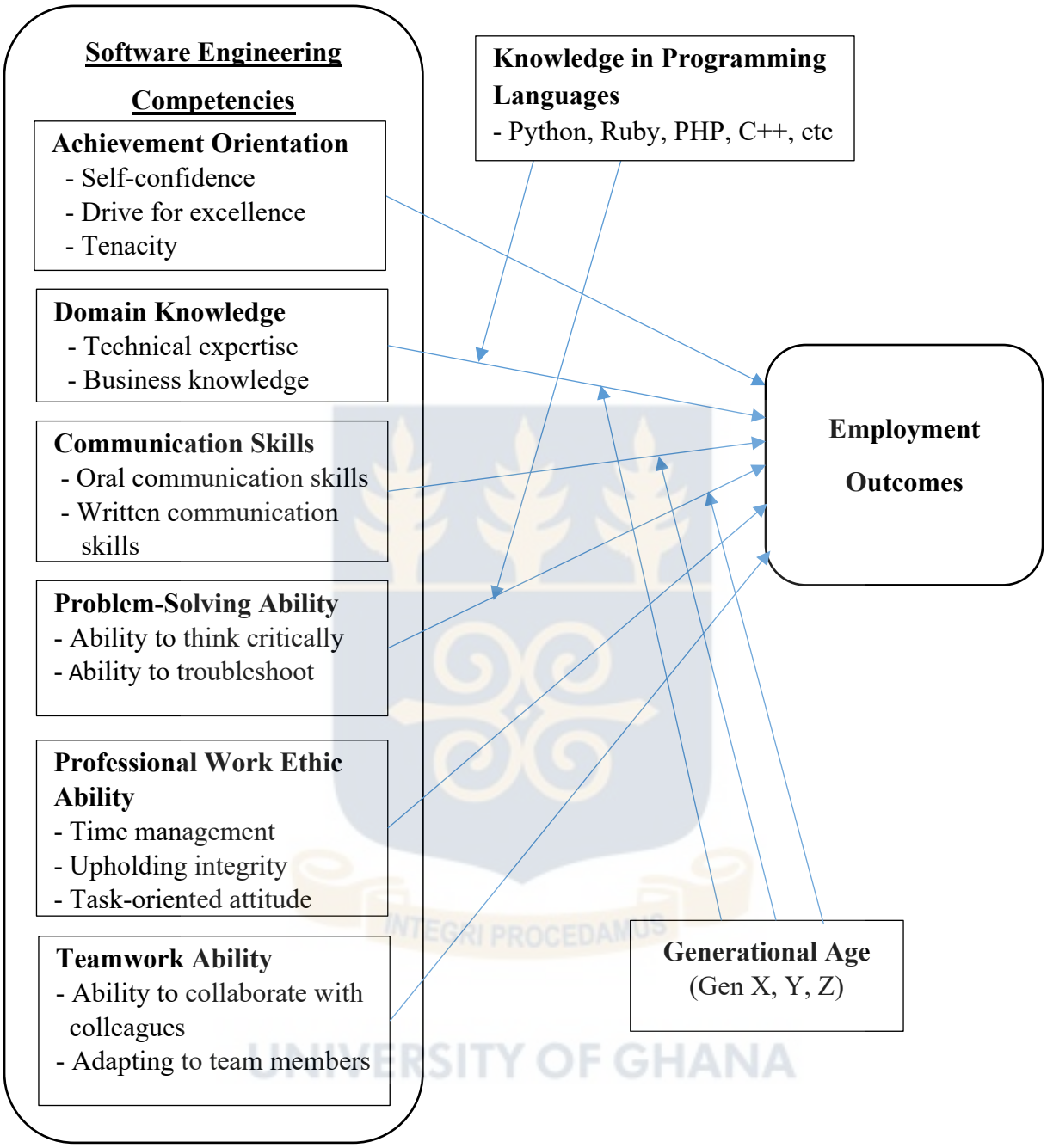

Figure 3.4 Conceptual Framework based on the Human Capital Theory

Source: Author's own constructs

3.4 Chapter Summary

In this chapter, the selection of the Human Capital Theory as the theoretical framework for this study is presented. Human Capital Theory underpins the investigation into how investing in oneself through education and training to acquire in-demand competencies lead to positive employment outcomes in the SE industry. The theory shape the exploration of how generational differences and human capital development influence the employment outcomes of software engineering professionals. The developed framework serves to guide data collection and analysis in subsequent chapters.

CHAPTER FOUR

RESEARCH METHODOLOGY

4.1 Chapter Overview

This chapter provides a comprehensive overview of the research methods employed in this study. It begins by discussing the philosophical underpinnings of the research, particularly the chosen research paradigm and its implications for the investigation. The chapter then delves into the specific methods used for data collection and analysis, the sample size, and the sampling technique used for the study. The data analysis methods, including thematic analysis and document analysis, are elaborated upon in detail. In essence, Chapter Four serves as a detailed roadmap for the research process, outlining each step taken to explore the influence of SEC on employment.

4.2 Research Paradigm

Research paradigms, also referred to as philosophical assumptions are essential for conducting well-grounded research. They serve as the underlying basis for the design and execution of research (Creswell, 2009). Paradigms dictate what the researcher aims to investigate, how they go about studying it, and how they interpret and comprehend knowledge. These paradigms encompass a collection of beliefs, values, and techniques shared within the scientific community and significantly impact research practices (Kuhn, 1970).

There are three paradigms in Information Systems (IS) research: Positivism, Interpretivism, and Critical Realism (Myers & Avison, 2002). Each of these paradigms provides a distinct viewpoint regarding the nature of reality, the relationship between researcher and subject, and the types of knowledge produced.

4.2.1 Positivism

The positivist paradigm proposes the belief that reality has static attributes, which are objective and quantifiable. It upholds the idea that all events in reality can undergo empirical inspection and allow for rational analysis through different methods (Leong, 2014). Positivist studies generally attempt to test theory to enhance the predictive understanding of phenomena (Saunders et al., 2003). This is accomplished using a deductive approach, where hypotheses are formulated based on existing theoretical knowledge and then put to the test through empirical observation (Cavana et al., 2001). The overarching goal of positivist research is to generate universally applicable predictions that can be utilized in a variety of settings.

4.2.2 Interpretivism

The Interpretivism paradigm, in contrast, holds that reality is socially constructed and subjective. It believes that people with different backgrounds, experiences, and assumptions contribute to the ongoing formation of reality existing in their broader social context through various social interactions (Wahyuni, 2012). Interpretivist researchers prefer to interact and have a dialogue with respondents to gain their interpretations of a phenomenon (Fisher, 2010). They often employ qualitative methods to explore the richness, depth, and complexity of phenomena.

4.2.3 Critical Realism

Critical realism acts as a bridge, connecting the epistemology of positivism with the ontology of interpretivism. It posits that even though our perception and societal constructs mold our interpretation of reality, there still exists an objective reality independent of our comprehension (Collier, 1994). Critical realism introduces the idea that there are certain incidents that remain unseen yet wield an impact on observable occurrences (Patomäki & Wight, 2000). By

employing a mix of quantitative and qualitative methods, critical realist researchers aim to examine and reveal the hidden mechanisms operating within a given context (Mingers et al., 2013). Through observations, surveys, or experiments, they gather empirical data. At the same time, however, they conduct qualitative analysis in an attempt to grasp the contextual factors as well as social processes that lend meaning and shape to these observed phenomena.

For this research, the **Interpretivism paradigm** has been chosen as the guiding lens. Interpretivism recognizes that research problems are found in social contexts and that the best approach to comprehending the behaviors of individuals may not always involve relying solely on statistical analyses. This paradigm originates from the belief that reality is shaped by human interactions (Fisher, 2010). This aligns with the objectives and problem of this study, which seeks to understand the phenomenon being studied in its social context through close interactions with participants (software engineers) to grasp their subjective viewpoints.

4.3 Research Design and Methods

Research design is a comprehensive plan, which allows research questions to be answered in a manner that is valid, objective, accurate, and economical (Trotman, 1996). It encompasses everything from general assumptions to specific strategies for collecting and analyzing data. The choice of research design depends on both the nature of the research question and the subject under investigation (Williams, 2007). Research designs generally fall into three categories: qualitative, quantitative, and mixed methods.

Qualitative research design focuses on gathering non-numerical information such as words, images, or objects. It aims to understand the perspectives and behaviors of individuals in

specific contexts (Creswell, 2013). This approach proves valuable when exploring complex issues or situations in great depth and detail.

Quantitative research design involves gathering numerical data that can be analyzed using statistical techniques (Creswell, 2009). It is commonly used when researchers want to measure how often something happens or determine quantities or frequencies involved in a phenomenon (Bloomfield & Fisher, 2019). This approach is especially useful when testing objective theories by examining connections between variables. The mixed methods approach on the other hand combines elements from both qualitative and quantitative approaches, to offer depth and breadth regarding a phenomenon under study (Creswell, 2013). This methodological approach delves into the underlying meanings and interpretations while also investigating relationships between variables. It is especially useful when a phenomenon needs to be explored before an instrument is administered, when statistical results need to be explained, or when comparing qualitative and quantitative findings to ensure consistency (Creswell, 2013).

In this study, a **qualitative research design** was adopted for the study. The selection of the qualitative approach was to allow for an exploration and nuanced understanding of the research questions as suggested by Creswell (2012). Thus, the qualitative approach is more aligned to the research objectives and purpose which seeks to understand and explore the phenomenon under study. Through the qualitative approach, a nuanced understanding can be ascertained regarding the complex relationship between SEC and employment, as well as the moderating effect of generational age and knowledge in programming languages.

4.4 Case Study as a Research Method

This study seeks to explore the impact of software engineering competencies on employment while also examining the moderating effects of generational age differences and proficiency in programming languages. With this in mind, a case study was deemed as an appropriate research method for the study.

A case study is commonly used in research to comprehensively examine a phenomenon or situation. This research method involves conducting an in-depth investigation of an individual, group or entity to gather data from different sources such as interviews, observations, and historical records (Gerring, 2004). Additionally, case studies allow for the exploration of intricate and diverse matters, offering valuable insights into the phenomenon under study (Yin, 2018). Furthermore, case studies allow researchers to study a particular phenomenon within its natural setting, providing rich and accurate information (Boateng, 2016).

According to Yin (2012), a case study research method is used to answer 'how' and 'why' research type of questions. The author describes three types of case studies: "(a) explanatory case studies, (b) descriptive case studies, and (c) exploratory case studies". Whereas explorative case studies are used for preliminary inquiry of a phenomenon, serving as a prelude to more in-depth inquiries, explanatory case studies are designed to explain how events occur by gathering data on cause-effect relationships. Descriptive case studies begin with a theory that describes the subject being studied, and then the information collected is compared to the developed theory (Yin, 2012). Although many social scientists believe that case studies should be confined to 'exploratory' studies, several scholars have used the case study research method for 'descriptive' and 'explanatory' studies (Yin, 2018).

Moreover, case studies can take the form of a single case study or a multiple case study. A single case study is used when a researcher seeks an in-depth study of one group, event, or person. It has the advantage of providing rich descriptions of the specific phenomenon under study (Yin, 2018). However, with single case studies, there is an inherent risk of misjudging the representativeness of the selected case, and a lack of generalizability of findings (Kahkonen, 2014). Conversely, a multiple case study otherwise known as a collective case study, involves an in-depth study of two or more groups, events, or persons. This makes multiple case studies more robust since they are supported by replication logic (Yin, 1998). Thus, since each case is treated as a separate inquiry, the findings obtained from a multiple case study strengthen the certainty of the results which end up either supporting or challenging an emerging theory (Amerson, 2011; Yin, 2003). Due to its benefits, the multiple case study was used for this study as the researcher sought to undertake independent case studies of the same phenomenon.

4.4.1 Case Study Design

This study uses the descriptive case study approach which involves the use of theory to guide data collection and subsequent confirmation or challenge of the theory used. In information systems research, the case study approach stands out as the most common qualitative method (Darke et al., 1998). The case study approach was selected for its adaptability and the multiple data sources it offers (Yin, 2018). The unit of analysis for the study is the individual software companies in Ghana that participated in the study. The basis for the selection is that software companies are the primary context within which the phenomenon of interest (SEC and their influence on employment) are observed and analyzed. Thus, software companies are the environments where essential competencies are defined and linked to employment outcomes.

4.4.2 Sampling Technique for Selecting Respondents

Sampling is a critical aspect of any research study. It involves the careful selection of a subset of individuals from a larger population to participate in the study (Lunsford & Lunsford, 1995). A good sample is one that represents the population well and produces results similar to those that would have been obtained if data had been collected from the entire population (Acharya et al., 2013). Samples are so selected in research to reduce the cost and time associated with undertaking research (Creswell, 2012).

There are two main classifications of sampling techniques: probability sampling and non-probability sampling (Creswell, 2012; Sharma, 2017). Probability sampling allows for reasonable statistical inferences as it ensures that each member of a population has an equal opportunity to be chosen for a study. This includes methods like simple random sampling or stratified sampling. On the contrary, non-probability sampling does not allow for valid population generalizations since it does not employ random selection and instead relies on the researcher's discretion or convenience in selecting participants (Alvi, 2016; Saunders et al., 2003).

Two non-probability sampling methods, convenience and purposive sampling, were used for the study. Convenience sampling is used when researchers select participants based on their easy accessibility and availability (Etikan, 2016). Thus, participants are chosen simply because they are convenient for the researcher to reach them or they are willing to participate. With purposive sampling, there is an intentional selection of participants based on their expertise in a subject matter or their specific characteristics that are deemed useful to the study (Etikan, 2016; Saunders et al., 2003).

In this study, the focus is to gather information on the relevant software engineering competencies required in Ghana, their influence on employment, and the moderating effect of knowledge in programming languages and generational age on the aforementioned relationship. This required the collection of data from top-level personnel of software companies in Ghana such as Chief Executive Officers, General Managers, Human Resource Managers, or Lead/Senior Developers. To achieve this, the researcher obtained a pool of over 10 reputable software companies to sample from. This was ascertained through a name search at the registrar general's website (with the aid of keywords such as 'software', and 'developer'), online Google searches, social media, and job portals in Ghana. Thereafter, company websites were accessed to gather email addresses, and invitations for participation in the study were then sent via email. The convenience sampling method was then utilized to select 2 companies located in Accra that were accessible and willing to partake in the study. Furthermore, the purposive sampling method was used to ensure the inclusion of participants who could provide valuable and relevant insights needed for the study. In total, six respondents were used for the study, with three from each of the two sampled companies.

4.4.3 Data Collection Methods

The data collection method for this study has been carefully designed to gather comprehensive, in-depth information to ascertain answers to the research questions posed. Several sources can be used to gather data for a case study. This includes interviews, direct observation, participant observation, archival records, documents, and physical artifacts (Yin, 2018).

The primary method of data collection for this study was interviews. However, according to Benbasat et al. (1987), collecting evidence for a case study requires that it come from two or more sources for the research findings to be supported. Therefore, document analysis and direct

observation were employed to provide support to the interviews conducted. This combined approach, aligning with Benbasat et al. (1987) emphasis on multiple sources, enhances triangulation, and provides a multi-faceted view of the research topic. The primary focus on interviews is in line with the interpretivist paradigm, aiming to understand the subjective experiences of key personnel within software companies. Document analysis and direct observation provided complementary data for cross-referencing and validation.

4.4.3.1 Interviews

The study used semi-structured interviews to collect first-hand data from respondents who were top-level personnel of software companies in Ghana. Interviews are a prominent means of data collection as it allows the interviewer to ask follow-up questions to gain a clear understanding of the interviewee's responses and to address any questions the interviewee may not have understood (Yin, 2003). The use of semi-structured interviews was to offer flexibility in questioning, allowing for the collection of rich and detailed information from respondents (Saunders et al., 2003). However, to guarantee uniformity throughout the interviews and keep the focus on the objectives of the study, a prepared interview guide was utilized. The interview guide was designed with the research questions in mind and informed by the interpretive paradigm's emphasis on exploring subjective experiences. To evaluate the effectiveness of the interview guide, a pilot interview with a relevant personnel in a software firm was undertaken before the main interviews.

The interview guide included both open-ended and close-ended questions. The open-ended questions were used to elicit detailed responses, and unique perspectives, and explore the rationale behind participants' opinions on the subject matter. Closed-ended questions were used in the study to collect pertinent demographic data from respondents. With the consent of

respondents, interviews were audio recorded to ensure the accuracy of collected data. In addition, notes were taken during the interviews to ensure documentation of key points and follow-up questions that may arise during the interview. This aligns with Yin's (2018) emphasis on capturing key information that contributes to meeting the study's overall objectives. In total, interviews were conducted at two software firms, with three top-level personnel from each company. Some of the interviews took place at the company premises, while others were conducted virtually via Google Meet due to respondents' busy schedules.

4.4.3.2 Ethical Considerations

This study adhered to strict ethical guidelines to ensure the integrity of the research as well as protect participants' interests. Firstly, all correspondence with the software firms was conducted using the university's official email to assure respondents of the study's academic nature and confidentiality. In addition, the interviews were conducted at the convenience of the respondent and maintained to a reasonable duration. Prior to the interview, the research objectives were explained to participants, and the academic aspect of the study was emphasized. The consent of respondents was also sought to audio-record the conservations. Finally, to encourage respondents to freely express their views, they were assured that the study would maintain their anonymity.

4.4.3.3 Document analysis

The websites of the software firms were analyzed to gather relevant information about the companies. These websites offered insights into the firms' histories, the software services that are offered, the software engineering positions available, and the qualifications and characteristics of the personnel employed. This provided the researcher with valuable insights into the phenomenon under study and aided in correlating findings from the interview.

4.4.3.4 Direct Observation

The researcher visited the premises of each of the two firms involved in the study. These visits provided valuable insights into how software firms operate, their personnel, and the overall work environment. Additionally, the visits offered a clearer understanding of the generational cohorts employed at these firms, which helped in correlating the responses given by the respondents.

4.5 Data Analysis and Interpretation

Analyzing gathered data and making meaningful interpretations is a crucial step in transforming raw qualitative data into meaningful insights (Mezmir, 2020; Miles & Huberman, 1994; Yin, 2018). Given the interpretivist philosophy underlying this study, thematic analysis was employed to systematically examine interview transcripts. Thematic analysis, as outlined by Miles and Huberman (1994), is well-suited for interpretivism research as it involves iterative data reduction and identification of patterns, which helps in drawing conclusions from the subjective experiences of respondents. Drawing from established qualitative techniques, the researcher remained receptive to discoveries throughout the data analysis to provide a detailed and insightful analysis and interpretation of the gathered data (Peel, 2020).

4.5.1 Thematic Analysis

Thematic analysis is a flexible qualitative method that involves identifying, analyzing, and interpreting patterns, known as themes, within interview transcripts (Vaismoradi et al., 2013). In using thematic analysis for the study, Miles and Huberman's (1994) approach was followed. The authors, Miles and Huberman (1994), outline four major steps in conducting thematic analysis: data collection, data display, data reduction, and conclusion drawing and verification.

The first step is data collection which was achieved through the conduct of interviews and subsequent transcription. The second step, data reduction, involves simplifying, abstracting, and transforming the collected data to find relevant accounts in the texts (Miles & Huberman, 1994). This is referred to by some scholars as data condensation, implying that collected data needs to be condensed to ensure large textual data is meaningfully reduced and organized (Mezmir, 2020). This step also involves thematic coding of relevant data. The third step, data display, requires making use of matrices, graphs, and charts, among others to visually display data. This makes it easier to make meanings that permit conclusion drawing (Lyons, 2000; Miles & Huberman, 1994). The fourth and final step involves making conclusions from the analyzed data. Drawing conclusions involves taking a step back to reflect on the meaning of the analyzed data and evaluating their significance in relation to the research questions. Verification, a crucial aspect in drawing conclusions, involves going back over the data as many times as needed to validate and confirm the emerging conclusions (Miles & Huberman, 1994). The process followed is shown in Figure 4.1.

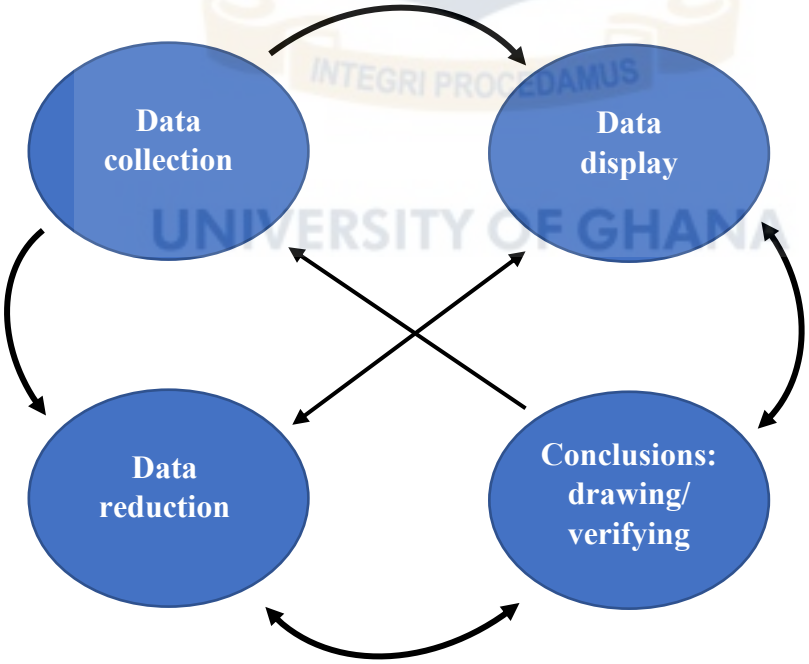

Figure 4.1 Miles and Huberman's Data Analysis

Source: Miles and Huberman (1994)

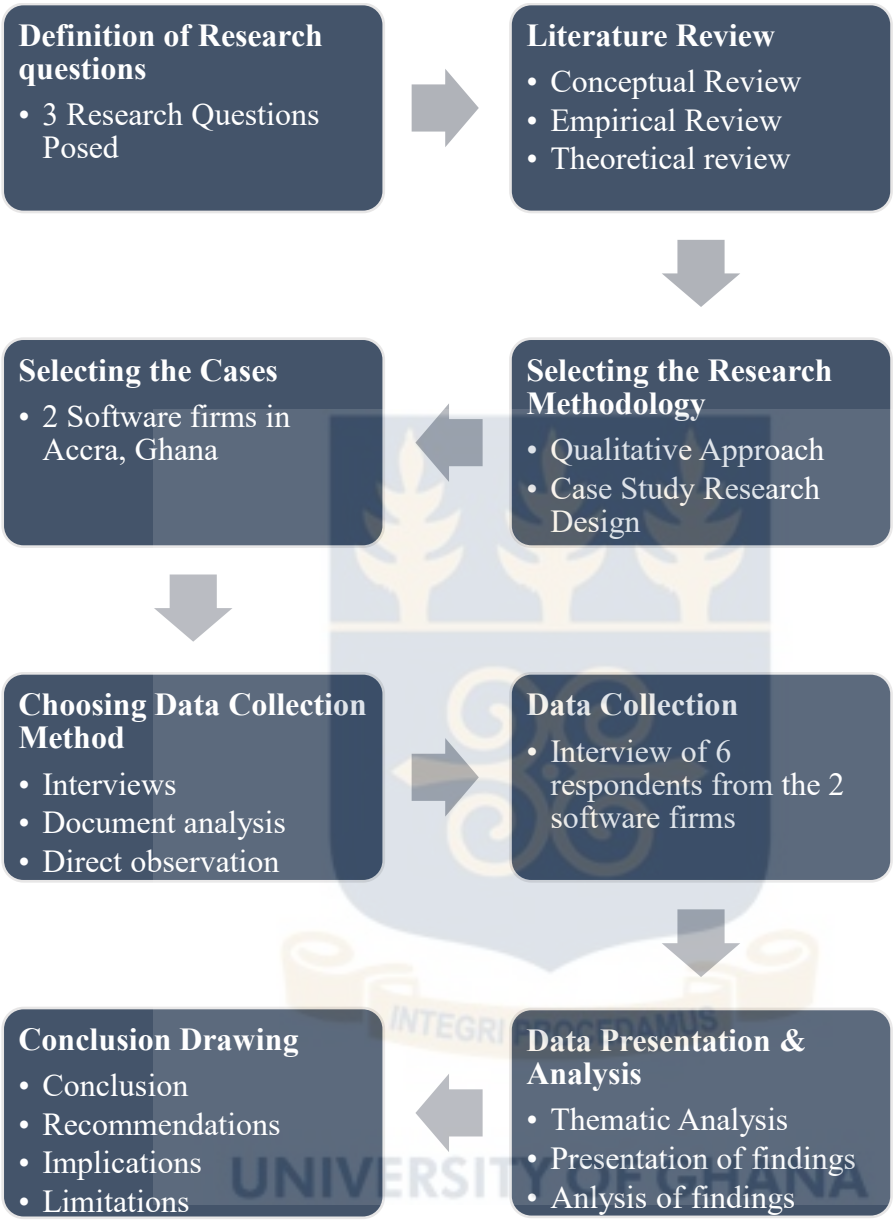

Figure 4.2 Flowchart of the Research Process

Source: Adapted from Kahkonen (2014)

4.6 Chapter Summary

The primary focus of this chapter was to present and discuss the research methodology used in addressing the research questions of this study. Careful deliberation was given towards selecting an appropriate research paradigm that could establish a solid philosophical foundation for this study. Subsequently, each step taken towards gathering and analyzing data was meticulously outlined to ensure comprehensiveness. The sampling technique was explained, highlighting the rationale for the convenient and purposive selection of participants. In terms of data collection, detailed explanations were provided regarding the use of interviews, document analysis, and direct observation. Additionally, the use of thematic analysis for data analysis was explained. Significantly, this comprehensive methodology establishes a robust framework for exploring the research questions. Therefore, this chapter provides a smooth transition into the presentation and analysis of collected data in subsequent chapters.

CHAPTER FIVE

RESEARCH FINDINGS

5.1 Chapter Overview

In the previous chapter, the methodologies used in undertaking this study were elaborated upon. This included the rationale behind the choice of the qualitative approach, the use of a multiple case study research design, and the use of a semi-structured interview as an instrument in collecting data from respondents. This chapter is a further step to achieving the objectives of the study by presenting findings from the two case studies that were undertaken during the data collection process. These findings will help in understanding the competencies required of software engineers in Ghana, as well as the influence of competencies on employment and the moderating influence of knowledge in programming languages and generational age.

5.2 Case Description

As explained in the previous chapter, this study employed a multiple case study approach by collecting data from two software firms in Ghana. This section provides descriptions of the two case studies. The first software firm is denoted as SwF 1, and the second as SwF 2 for the sake of anonymity.

5.2.1 SwF 1

SwF 1 is a software development company that has been in operational existence for over a decade. The company is duly registered with the registrar general and is located in the capital city of Ghana, Accra. Since its inception, the firm has experienced rapid and continuous development. The company boasts of more than 10 employees who occupy various positions in the firm. Some of the SE positions that exist in the company include Backend developers, Mobile developers, Front-end developers, UI/UX, and Quality assurance testers. Aside from

these SE positions, the firm has 5 board of directors, and other technical leads such as the Head of engineering, Head of DevOps, Programs manager, and Senior Software Engineer among others. SwF1 operates primarily as a financial technology (FinTech) software firm, providing innovative software solutions to meet the specific requirements of local and international clients.

5.2.2 Services Provided

As a bespoke software firm, SwF 1 is into the development of a full range of software products tailored to meet the needs of its clients. This includes the development of core applications such as Enterprise Applications, Client Relationship Management (CRM) systems, and Human Resource Management Systems (HRMS) among others. The firm also engages in FinTech, USSD services, and Web and Mobile applications development. However, SwF 1 is primarily FinTech and aside from developing custom financial solutions for clients, it also boasts of a collection of in-house software solutions that are designed to meet the requirements of banks, credit unions, savings and loans, and fund management firms, among others.

5.2.3 SwF 2

SwF 2 is a software development company located in the Capital city of Ghana, Accra. The firm was founded over a decade ago and boasts an impressive portfolio of software products. Just like the first case, SwF 2 has over 10 employees who play various roles within the company. Some of the software engineering positions that exist within the company include Head of Software Development, Head of Mobile App Development, Lead Web App developer, Chief Security Officer, Head of Networking, Project Manager, Data Analysts, and Lead Product Manager, among others. The firm provides development of innovative software solutions to the Ghanaian local market and beyond the borders of the country. To achieve such,

the firm constantly seeks individuals who have the requisite competencies and an inner drive to build incredible products. The company does not only attract the best software engineering talents, it has over the years trained a lot of persons, including novices, to become software engineers as highlighted by the co-founder in the interview.

5.2.4 Services Provided

The company specializes in web development, mobile application development, Internet of Things (IoT), systems and processes automation, artificial intelligence (AI), and IT consultancy, among others. According to the co-founder, the firm is yet to implement its Blockchain technology, though they have started something as such in the area of land administration. Aside from developing custom software solutions for clients, the firm also has in-house software products that are made available for subscription by organizations to improve their processes.

5.3 Demographic Characteristics of Participants

Overall, this study had 6 participants with 3 from each of the two software firms that were used in the case study. The demographic characteristics of the participants are indicated in Table 5.1.

Table 5.1 Demographic Characteristics of Participants

Software Firm	Employment Position	Role in Organization	Highest Educational Qualification	Software Development Experience (years)	Experience in managing developers (years)
SwF 1	Head of Engineering	• Leading software engineers • Backend engineer (C# developer)	Bachelor's degree in Computer Science	7	1
SwF 1	Program Manager	• Overseeing the activities of the other developers • Spearhead the development of software products • Educating and mentoring developers	Bachelor's degree in Computer Science and Mathematics	8	3
SwF 1	Senior Software Engineer	• Maintaining of developed software • Handling integrations with partners such as MTN, AirtelTigo, among others. • Main go-to developer for support for mobile banking clients	Diploma	6	3

SwF 2	Co-founder / Project Manager	• Managing software projects • Handles software security • Testing software • Checking for bugs and vulnerabilities in the application development process.	Master's degree in Computer Science	14	8
SwF 2	Head of Software Development	• Managing developers • Handles development of web applications	Bachelor of Science, Physics	8	3
SwF 2	Head of Mobile App Development	• Managing other developers • Overseeing the development of mobile applications.	HND in Computer Science	4	2

Source: Author's construct based on findings

5.4 Findings from the Case

The findings from the two case studies conducted are presented in this section. This is a further step in achieving the objectives of the study which are to examine the required SEC, the influence of SEC on employment outcomes, and the moderating influence of knowledge in programming languages and generational age.

5.4.1 Case one: Software Engineering Competencies (SEC) required by Employers in Ghana (SwF 1)

The software engineering competencies gathered from case one (SwF 1) are presented in Table 5.2. In the table, the Head of Engineering is denoted as HoE, the Program Manager as PgM, and the Senior Software Engineer as SSE.

Table 5.2 Software Engineering Competencies, Description, and Importance (Case 1)

Technical Skills	Description	Reason of Importance
1. Proficiency in the relevant programming language (C#, Flutter, React, Golang, etc)	"So the programming language is the language or the tool that you use to develop a system that will be used." *(HoE)*	• "to be a software engineer, you should know the tool that is used to work" *(HoE)* • "we expect everybody to know how to use the C# programming language because about 90% of all the company's software is built in C#" *(SSE)*
2. Version control (GitHub)	• "It's more like a history for your code" *(PgM)* • "That is the place where you push your code." *(HoE)* • "the GIT is just like a place where we merge	• "it helps you to actually keep track of the changes that you have made in your code base, the branches created, pull requests made, the workflows that are in place, pipelines, and all those things." *(PgM)*

	[various modules of an application]" *(HoE)*	• "So we put the software together, then on the GIT we have UAT [Unit Acceptance Testing], that is a test environment where we can run and be sure that all that we asked engineer A or engineer B, he has done what is required of him. So the GIT is actually for testing purposes." *(HoE)*
3. Database Management	• "it is a storage where your data goes in and comes out." *(HoE)* • "it is like an API, an interface or a middleware" *(HoE)* • So without the database, you are just getting the information and cannot store it anywhere. It is like a storehouse where you store data. *(HoE)*	• "majority of what we do is building banking systems, and the core of the banking systems is actually the database. So we expect people to know how to work with databases, how to write SQL queries, how to back up databases and also ensure data integrity." *(SSE)* • "it is important because around FinTech we have a lot of regulations. So there are a lot of regulations that the Bank of Ghana has set in regard to data management and data integrity. So because FinTech would not want to risk that, we would expect that at least candidates know how to work with the databases, and know how to ensure data integrity." *(SSE)* • "There's something we call race conditions, where we would have the same data being overwritten.

		So it leads to wrong data, essentially. So we want to avoid that in the banking system. So we at least, have to ensure that everyone at least knows how to work with a database." *(SSE)*
4. Software Development Life Cycle	• "They come in the form of processes, you move from one stage to the other. There is waterfall, then agile where you can start a project and then make improvements in a cycle form, like when you plan, you design, you develop, you test, you deploy, you maintain, and then you come back to design, develop, test and maintain and so on until you get a near perfect application." *(PgM)*	• "You should also have an understanding of the development methodologies that we use, whether it is Agile, Scrum, Kanban, and so on." *(PgM)*
5. Data structures and algorithms	• "these things come with your understanding of the data structures, like arrays. How do you even create an array?" *(PgM)*	• "So if I do not know how to create an array, I don't know how to create a linked list, I don't know how to create a graph, it will be hard for me to even build an algorithm that is going to solve a complex problem." *(PgM)* • "you need those skills like it's what shows that you have a good understanding of software

		development, aside from the language." *(PgM)*
6. Project management platforms such as Jira	• "so aside we having a history for our code base, we should also have a history of the task that you have been assigned to. So when you go to a project management platform like Jira, normally your team lead or your immediate supervisor will assign you task over there" *(PgM)*	• "So I don't have to always come to your desk and ask you how far, where are you? What have you done?" *(PgM)* • "That is a way we can use to keep track of the progress of the project so that we can report to management, or whoever the stakeholder that is interested in that data." *(PgM)*
7. Testing and integration	• "we have something called unit test, where you test each of the components of the application that is built." *(PgM)* • "the integration part has to do with after you've written your test, how does it integrate with the rest of the application or the rest of the system." *(PgM)*	• "Testing and integration is very key. As a developer, you should be able to know how to write unit tests." *(PgM)* • "So I'm able to test each and every component of the application, know that it's working, and then I now bundle it, and then I ship it to the customers to run their tests, or to the QA team to run their tests before we ship to our customers." *(PgM)*
8. Communication	• "We know we must all understand the same thing before we can even be in one place." *(HoE)* • "in every organization, you should have clear cut hierarchy of	• "So it's important because our customers are banks, essentially. So if you as a developer is the lead on a particular project, we would be expecting you to talk to the customers who are the banks. These are bank CEOs, bank

	communication routes that you maintain within that organization" *(PgM)*	managers, and maybe the head of engineering for that particular bank. So we would expect that you can be able to communicate your thoughts clearly and precisely to the customers." *(SSE)*
9. Interpersonal skills	• "I think this cuts across everywhere. In any organization, the way you relate with your boss, your colleagues, you should just know what to say at what time." *(HoE)*	• "Either than that, you will be having issues with people." *(HoE)* • "So I would then have to go and interact with the person and know why there is that gap, and see how best we can bridge it." *(PgM)*
10. Mindset for growth and learning	• "You may know that let's say today, how to get a person's username is to write a.username, tomorrow there may be a more efficient way of doing the same thing." *(HoE)*	• "As an engineer, you should have the mindset of growth and learning, because the space where you find yourself, is a fast-growing industry." *(PgM)* • "So if you are always ready to learn, you will improve your skill and you will always be up to date." *(HoE)*
11. Problem solving	• "Problem-solving is like the basis of everything that you do. You're always given a problem to solve." *(PgM)*	• "I should be able to think through all of this and break them down into micro-tasks that I can work on within the feature that has been assigned to me" *(PgM)* • "especially, and it should also be in line with the requirements that we've gathered from the customer." *(PgM)*

12 Collaboration	• "As I mentioned with the example with the GIT and then the Jira, you should be able to collaborate as a team. In most cases you will be working as a team," *(PgM)*	• "if you go alone, you cannot do so much. But if you go with other people, you can reach far" *(PgM)* • "So a back-end developer depends on the database guy to be able to get queries and schemas in place" *(PgM)*
13. Adapt to new environments	• "you should be able to adapt to that environment, the culture of the company, the do's and don'ts" *(PgM)*	• "you should be able to adapt to that environment and the technologies that they use there." *(PgM)* • "you should be able to work with all of your co-workers, those who like you, those who don't." *(PgM)*
14. Time management	• "How soon are you able to close tasks that have been given to you" *(PgM)*	• "So time management is very key, it helps you to meet deadlines without sacrificing quality" *(PgM)* • "Time management is very key when it comes to calculating your KPI [key performance indicator]" *(PgM)*
15. Leadership skills	• "Leadership skills has to do with those who are team leads, those who are managers like me." *(PgM)* • "We should be able to inspire and motivate them when they are feeling down." *(PgM)*	• "our actions tend to influence other people ….. so we should be careful the way we handle things that occur at the office, like conflicts." *(PgM)*
16. Being ethical	• "being ethical in your activities means that the way you do your work	• "See how your actions are affecting other people…. there are some people, they tend to play a lot when they're at the office and

	shouldn't badly affect another person." *(PgM)*	it's distracting to other people" *(PgM)*
17. Self-motivation	• "you should be able to motivate yourself to learn that language, to keep your employment, to add to your skill set" *(PgM)*	• "you need to motivate yourself, else you cannot keep up and you take offense and you might even resign." *(PgM)*
18. Prior work experience	• "we would expect a potential employee to have worked elsewhere" *(SSE)*	• "unless it's a very junior role, an internship role, we would expect a potential employee to have worked elsewhere so that they are already familiar with the industry and know how to work with their colleagues" *(SSE)*

Source: Author's construct based on findings

5.4.2 Achievement Orientation and Employment

This section discusses the importance of achievement orientation to software firms (SwF 1) and how such competency influences employment outcomes in software engineering.

The Senior Software Engineer said:

> "*In our organization, I mean, not just ours, but any other FinTech that you would think about, it's always a very high pressured environment. Everyone else does not go to work on Saturdays and Sundays, but on Saturdays and Sundays, people are using their banking applications. They are using their banks, using their ATM cards. They are using their mobile banking apps. So in FinTech, most of us don't really sleep, so we*

sort of assess you based on how you can handle stress. And how determined you are to work under pressure." *(Case 1 – Senior Software Engineer)*

The Senior Software Engineer added that:

"*especially when it comes to hiring junior developers, that's something that even I, personally, tend to look out for. Is this person really hungry for what he wants to do? So yes, the drive for excellence, confidence in one's self, is what we tend to look out for, especially in junior developers.*" *(Case 1 – Senior Software Engineer)*

The Program Manager also said:

"In the context of what I was describing that the language [programming language] changed, you should be unfazed by things like that. You should be willing to learn, and bring out your best to be able to achieve the tasks, like to be able to achieve completeness. When it comes to fulfilling your tasks, you should be able to challenge yourself and take on new technologies within the workspace." (Case 1 – Program Manager)

The Head of Engineering also added:

"*I can even link it [achievement orientation] to continuous learning. You need self-confidence to learn. So if you are not learning you will be outdated. If new technologies are out because you are not always abreast with trending things, you will be left behind, and if care is not taken, you can lose your job.*" *(Case 1 – Head of Engineering)*

The responses given by the respondents indicate that achievement orientation is a competency software firms are very mindful of. Within the software engineering industry, there is a constant need to update one's skillset due to rapid technological advancements (Oguz & Oguz, 2019).

Hence, there is a need for software engineers to possess self-confidence, tenacity, and drive in order to keep pace with any new trends or technological changes that may emerge (Calazans et al., 2017). In summary, the respondents consider achievement orientation as an important competency that influences the employment outcomes of software engineers.

5.4.3 Domain Knowledge and Employment

This section presents the findings of Case 1 concerning the importance of domain knowledge and whether it influences the employment outcomes of software engineers. The respondents were convergent in their opinions on domain knowledge.

The Head of Engineering said:

> "*It is important because before you will be able to build a FinTech application, a financial system, you should know a little about finance. You should know your Debit and Credit. So if you don't have that knowledge, you can't build any system. Either than that the system that you will build, people will just be withdrawing money freely.*" *(Case 1 – Head of Engineering)*

He added that:

> "*Yes, it does affect employment. The domain knowledge is even the core. That's the core. If you don't have that knowledge we can't work.*"

The Program Manager also said:

> "*It's very important. So I'm a mobile application developer for instance. I build applications using Flutter, how active am I in the Flutter community? How up-to-date am I with their new releases, events, and things that they are actually working on? How*

updated am I with that information? So we need to know because sometimes they improve on those things." *(Case 1 – Program Manager)*

He added that:

"*I think in most cases it actually matters, because let's say you're building an application for logistics, you should be able to understand what happens in logistics. You're building an application for a hospital, you should be able to understand what happens in the healthcare industry. So it is very key, or maybe you are building a betting platform and you don't know anything about betting.*"

The Senior Software Engineer also added:

"*If you're working on the core banking team, a strong understanding of accounting is something that's really required. It's [domain knowledge] also required because if you are working on the core banking team, you are the one who is going to take support requests and feature requests from the customers, which are banks. So you have to be able to understand the business side of things, to be able to understand how the customers work.*" *(Case 1 – Senior Software Engineer)*

From the responses given, it can be deduced that domain knowledge is a key competency that software firms look out for. One must possess extensive business and technical knowledge in his area of expertise (Jebreen & Nabot, 2021). The FinTech niche in particular requires individuals who aside the technical expertise, understand the business aspects as well (Gupta et al., 2024). In summary, domain knowledge is a very important competency and as such influences the employment outcomes of software engineers.

5.4.4 Communication Skills and Employment

This section addresses the importance of communication for software firms (SwF 1) and how it influences employment outcomes in software engineers. Communication appears to be a very important competency to software firms though its impact on employment is limited.

The Program Manger said:

> "*For communication, it's important for any professional within a working space. You should be able to communicate effectively and efficiently as well. Your communication should be able to convey the information that you want to share with the recipient. So you should aim at making sure that whatever you're trying to tell the other person sits well with the person. The person understands what you are asking them to do or what you are requesting from them, and they are able to give you the appropriate feedback.*" *(Case 1 – Program Manager)*

The Senior Software Engineer also added:

> "*Yes, it's [communication] quite important, because as I said, you could be on a team that is building an app for one of the sponsors. So if you are the developer who is leading that particular project, you have to be able to effectively communicate between the sponsors and the rest of the engineering team. So you'll have to be able to break things down into human-readable chunks.*" *(Case 1 – Senior Software Engineer)*

He added that:

> "*A lot of developers are really technical. They really find it hard to convert that technical speak into a more human friendly speak. So it's quite important for a lot of developers to be able to express themselves.*" *(Case 1 – Senior Software Engineer)*

However, on whether communication affects the employment outcomes of software engineers, the Senior Software Engineer said:

> "*Not 100%. Because, as I said, some of the developers that we might be expecting are interns, junior developers, so we don't expect them to have that [communication skills]. Probably they might be really good, but because it's probably their first role, their first job, we sort of understand that.*" *(Case 1 – Senior Software Engineer)*

The Head of Engineering also said:

> "*It is [important], but it's 50 50. As I said, someone can be able to express themselves very well, but you give the person the work, the person doesn't know. That's why we have probation, when we realize that you just came to deceive us, but you don't really know what you're about.*" *(Case 1 – Head of Engineering)*

As to whether communication affects employment outcomes, the Head of Engineering said:

> "*I will say yes no because even on the sheet [interview scorecard] it is 10 marks. So you can get 0 there but the domain, you are very good. So it is 50 50.*" *(Case 1 – Head of Engineering)*

From the responses given, it can be deduced that communication skills is important to software firms. It enables software engineers to interact effectively and efficiently with project members, superiors, as well as project sponsors (Groeneveld et al., 2021). It also allows them to convey technical messages in a much more friendly language to sponsors and other stakeholders of a software project (Nguyen, 2020). However, the responses indicate that although communication skills is important to software firms, its impact on employment outcomes is

limited. In summary, communication skills is considered important to software firms but its influence on employment outcomes is limited.

5.4.5 Problem-Solving Ability and Employment

This section sought to examine how important problem-solving ability is to software firms and how it impacts the employment outcomes of software engineers.

The Head of Engineering said:

> "*Problem-solving is very key. Because what we are doing is solving problems. We develop systems to solve problems. So your logical reasoning, that one is very key.*" *(Case 1 – Head of Engineering)*

He added that:

> "*I remember in one of the interviews, I asked them let's assume we have two variables, maybe a = 19, b = 20, and I just want to interchange the value, it's problem-solving. How fast you can think when you face a problem so that one [problem-solving ability] too is very key.*" *(Case 1 – Head of Engineering)*

The Senior Software Engineer also added:

> "*That's very important. I mean, when we test somebody during a software engineering interview, that's usually what we're testing for. So we'd usually have like two or three calls. The first call is sort of like an introductory call to get to know the candidates. The second call is a more technical call where we sort of test the candidate's problem-solving skills. Because in our industry, in as much as we are building cool and fantastic applications, the core of it is all about problem solving. So we are all about testing the*

candidates on their problem-solving skills. So it's actually really important. That's something we do upfront, and then we validate it during the probation period." *(Case 1 – Senior Software Engineer)*

The Program Manager also added:

"*So you should be able to think analytically. When given a task, you should be able to break them down into micro tasks. Let's say I've been given a feature to work on, an authentication feature in an application. What goes into authentication on a platform? Well, a user would want to sign in, and a user who doesn't have an account with us would want to sign up. A user who has forgotten their credentials would want to reset their credentials one way or the other. So I should be able to think through all of this and break them down into micro tasks that I can work on within the feature that has been assigned to me, and it should also be in line with the requirements that we have gathered from the customer*." *(Case 1 – Program Manager)*

Hence the above findings clearly indicate that problem-solving ability is a competency that is highly sought after by software firms. Thus, software engineering is all about solving problems, hence engineers who are analytical and can solve problems are sought after (Stamm, 2023). In summary, problem-solving skills is very crucial for software firms and thus influence the employment outcomes of software engineers.

5.4.6 Professional Work Ethic Ability and Employment

This section deals with how professional work ethic ability such as such as integrity, self-discipline, hard work, time management, and task-oriented attitude influence employment outcomes of software engineers. The respondents pointed out that this competency is very

important. However, it's quite difficult to assess it during the interview stage, hence it is rather evaluated during the probation period.

The Program Manager said

> "*It's important, especially when we have to meet timelines. If you have been given a task that you are supposed to complete in four hours, and it has taken you three days because probably you are engaged with other clients. There are people who come to work and they work for other people with your own resources as a company, it is not nice to do that. So we want to know, when you are working for us and we have given you a task and a time frame, if it is unrealistic to complete that task within that time frame, you can communicate it to us, and be professional about it.*" *(Case 1 – Program Manager)*

He added that:

> "*So it will actually affect the employment opportunity that we give to you. If you are on probation and then we realize when you come to work, you don't do your work, you wait till like, three, four [pm] before you start doing the work and then it will look like you are not able to finish, so the next day you will continue it and it's consistent we will get to know of it, and then we can terminate your employment.*" *(Case 1 – Program Manager)*

The Head of Engineering also said:

> "*So let's say I'm given task A to finish, and I have a deadline to meet, and I have not said anything, it can affect my employment. Because the work that we do, it is time bound, and you are supposed to deliver on time. Even if there is any obstacle, you are*

supposed to share so that even if possible, the whole team will brainstorm and find the best possible solution." *(Case 1 – Head of Engineering)*

He added that:

"*Yes, [it does affect employment outcomes] at probation but not the initial [recruitment] stage. And also timing, maybe you are supposed to arrive at work at maybe 8:30 [am], and you know for developers we don't joke with our time. That is one thing. If they say closing time is 5[pm], even if the person is doing serious work when it's 5[pm], they will put their bags down and go. Yes, some of them that is how they are, especially the new guys that are coming, they put their bags down and go. So if you are obeying your closing time, then you should also be ready to come early, right? And if they go extra, you have to pay overtime.*" *(Case 1 – Head of Engineering)*

The Senior Software Engineer also added:

"*So within our organization, we have a probation period which can range from three to six months. So within that time, we sort of assess whether the candidate is managing their time effectively, comes to work early, has improved a lot, and can now communicate efficiently. So within that time, we sort of assess all of those stuffs. So, it is not important when you are coming in the door regardless of your level of experience. But during the probation period, that is when we sort of check it to make sure that you are sort of aligned with the company's goals, and you are also doing well at your job.*" *(Case 1 – Senior Software Engineer)*

Therefore, it can be concluded from the above comments that software firms value professional work ethic ability, even though it is mostly evaluated during the probationary period. This is particularly important since software firms have to meet stringent timelines from their clients

(Meyer et al., 2019), which increases the value of qualities such as self-discipline, hard work, time management, and a task-oriented attitude. In summary, having a strong professional work ethic is important for software firms and thus affects the employment outcomes of software engineers.

5.4.7 Teamwork Ability and Employment

This section discusses teamwork ability, its importance, and its influence on the employment outcomes of software engineers. The responses suggest that teamwork ability is an important competence although its effect on employment outcomes depends on the role in question.

With respect to teamwork ability, the Program Manager said:

> "*As the saying goes, if you go alone you cannot do so much, but if you go with other people, you can reach far. So teamwork is very key. One way or the other, you will rely on other people to be able to complete your day-to-day activities.*" *(Case 1 – Program Manager)*

The Senior Software Engineer also added:

> "*In as much as we want people to be able to work effectively in the team, we also want them to be their own, let's just say their own bosses. They know how to work independently without bugging everyone around. So we want people who know how to take charge of whatever it is that is given to them and be an adult about what they are doing. So that's primary. But then teamwork, yes, secondary, because there are times when the entire team has to work together to sort of get a particular product out quickly. We want the person to be a team player who can ask questions and others can also*

come to them to ask questions and resolve an issue quickly." *(Case 1 – Senior Software Engineer)*

He added that:

"*So it depends on the role. If it is a junior role, probably we will test you based on what you can do by yourself, and then validate you during the probation period. But for senior roles, we will sort of validate you [on your teamwork ability] using your prior work experience.*" *(Case 1 – Senior Software Engineer)*

The Head of Engineering also said:

"*You are coming to work with human beings, as I said with your colleagues. Even a project such as a school management system, will have two people or three people handling that. So you should know how to work in teams, and it's very important.*" *(Case 1 – Head of Engineering)*

He added that:

"*At the initial [recruitment] stages not really. If you are ready to learn and work in teams, I cannot say definitely that it affects [employment outcomes] but it depends on the organization. But here when you come, you will learn because you will be working with a project team.*" *(Case 1 – Head of Engineering)*

Hence, it can be deduced from the responses given that teamwork ability is an important competence especially given the collaborative environments of software engineering (Jia et al., 2017; Soomro et al., 2016). However, the responses also show that although teamwork ability is an important competence, software firms usually do not evaluate it during their hiring

process. They consider teamwork as a competence that can be nurtured in their employees by placing them in project teams for example. In addition, the responses indicate that ability to work independently may prove rather more important for junior roles as opposed to teamwork which rather proves more important for middle and senior roles. In summary, teamwork ability is important to software firms and does affect employment outcomes, although the extent of its influence depends on the role. The ability to work independently is preferred for junior roles while teamwork is preferred for mid to senior roles.

5.4.8 Impact of Knowledge in Programming Languages

This aspect presents findings of the moderating influence of knowledge in programming languages on domain knowledge and problem-solving abilities of software engineers.

5.4.8.1 Impact of Knowledge in Programming Languages on Domain Knowledge and Employment.

These responses were as follows:

The Head of Engineering said:

> "*Domain knowledge is centered around programming languages, so it affects everything. That's the key. So when someone is good at programming languages, that person will be well versed in domain knowledge.*" *(Case 1 – Head of Engineering)*

He added that:

> "*Let me use myself as an example, when I came to this company I was a C# developer but I didn't know much about it, I was new. But it got to a time that they wanted someone who can do front-end. But because I've done something [concerning front-end], I quickly moved there. So how this can affect you employment is when maybe the*

company is running out of assets and they need people to do multiple stuffs because they do not have enough to pay everyone."

The Program Manager also added:

"*So the thing is if you are proficient in your programming language, and you also have an understanding of your domain, like the FinTech that I'm in right now, it helps you to build solutions that are fine-tuned to solve problems that are actually faced within the industry or with clients that are actually going to use your application.*" *(Case 1 – Program Manager)*

Hence, from the responses given, it can be said that when software engineers are proficient in programming languages, it helps them to understand their respective domains better (Cheng, 2018). This enables them to build solutions that are fine-tuned to solve problems that confront the respective industry which will lead to positive employment outcomes. In summary, knowledge in programming languages moderates the relationship between domain knowledge and employment.

5.4.8.2 Impact of Knowledge in Programming Languages on Problem-Solving Ability and Employment

The Head of Engineering said:

"*I will say that it doesn't enhance your problem-solving [abilities]. When you work on more projects, that's how you [develop problem-solving ability]. Let's say someone who knows C#, knows Java, it doesn't enhance your problem solving but it's the projects that you build that enhance your problem-solving.*" *(Case 1 – Head of Engineering)*

He added that:

> "*So in school they taught us C++, VB [Visual Basic], but I have not really worked with any project on it, so it won't enhance my problem solving. But when I work on projects, then it will enhance my problem-solving. So I work on projects in C#, I work on another project in React, so this is how [you improve your problem-solving abilities].*" *(Case 1 – Head of Engineering)*

The Senior Software Engineer also added:

> "*I would say there is no direct correlation between that [between knowledge in programming languages and problem-solving abilities]. It is like how we all speak English, but certain people are very good negotiators. They had to learn that particular skill, right? And certain languages are built for certain problems, so as you learn more languages, it will help you just a bit over there. But if you want to actually improve your problem-solving skills, you have to actually take on challenges.*" *(Case 1 – Senior Software Engineer)*

From the responses given, it is clear that possessing knowledge in programming languages does not enhance one's problem-solving abilities. Problem-solving abilities of software engineers are rather enhanced when they execute software projects or when they take on challenges (Dekhane et al., 2013). In summary, knowledge in programming languages does not moderate the relationship between problem-solving ability and employment.

5.4.9 Impact of Generational Age

This section presents findings on whether generational age moderates the relationship between certain competencies and employment.

5.4.9.1 Impact of Generational Age on Domain Knowledge and Employment

This section examined whether there are differences in domain knowledge across the various generational cohorts (Gen X, Y, and Z) and whether they affect employment outcomes. The following were responses from the respondents:

The Head of Engineering said:

> "*I think yes [there are differences], because the X [Gen X], it's like what they know, changing is difficult. They learned the thing the hard way. But in the Y [Gen Y], at least there is a much clearer understanding and there is little flexibility. And even the Z [Gen Z], most of them have much knowledge [domain knowledge] because they are exposed to the thing at a very tender age, so they understand the concept better. But the knowledge [domain knowledge] is the X, they know the thing, but the implementation is the Y and Z, they have that.*" *(Case 1 – Head of Engineering)*

He added that:

> "*For software engineering firms, a lot of them want young people. One thing they also look at is not married. Because this our work, they want someone that can be dedicated. Because there are times that you have errors, like issues that you cannot even sleep. Like, even when you are on your bed, you are thinking, and when the thing comes up, you want to jump up and resolve the issue. But if you have a lot of responsibilities, if you close here, you have closed.*"

The Senior Software Engineer also added:

> "*The main differences [in domain knowledge] is with Gen Z, because they are adopting more of recent tech, that is in the space of AI. So what I see is that in the next 10 years, one of the requirements when we are talking about technical skills that will be required for developers, AI is going to be one of them. So because the AI revolution is already starting, and right now, personally, if I am to be hiring, I would like to see someone who really understands AI, because it has really taken us by storm. So that is something I would say, that when it comes to domain knowledge, Gen Z is quite excelling at that, because they are the ones mostly working with AI right now.*" *(Case 1 – Senior Software Engineer)*

The Program Manager also said:

> "*Yes, a lot [there are lot of differences in domain knowledge]. Especially those in the older generation, the X and Y, they possess a lot of knowledge in the domain because of experience. Like, they work with a lot of clients so they possess a lot of knowledge [domain knowledge]. A lot of them have certifications in most of these things, masters, some even have PhDs and so on. So they possess a lot of knowledge [domain knowledge].*" *(Case 1 – Program Manager)*

He added that:

> "*So let's say we want to employ someone in Gen X, I'm not sure we will employ the person to come and write code. We will employ the person to come and share domain knowledge and experience. So that is where you find a person like that being in a top*

management role. When you employ someone in Gen Y, they will either be in a top management role, or most likely be a CTO, or maybe a project manager."

From the responses, it is clear that older generations (Gen X and Y) possess extensive domain knowledge whereas Gen Z who are the youngest members of the workforce are abreast with recent technologies such as AI. In addition, Gen X are preferred for coding (programming) roles whereas older generations (Gen X and Y) are preferred for management positions. In summary, there are differences in domain knowledge between the various generational cohorts (Gen X, Y, and Z) which in turn affects the employment outcomes of software engineers.

5.4.9.2 Impact of Generational Age on Communication and Employment

This section sought to examine whether the various generational cohorts exhibit different communication skills, and if such affects the employment outcomes of software engineers.

The Program Manager said:

"*They [Gen X] want things to be more formal; mostly emails, so they are mostly formal. For the Gen Y and Z, we like to use these platforms; Slack, Skype, WhatsApp, we want those things. So our mode of communication is mostly playful, jovial.*" *(Case 1 – Program Manager)*

The Head of Engineering also said:

"*Yes, you see the Gen X, they are discipline. Like they know what to say at what time. Because at their time, they enforced those things [discipline], but the Z, whatever comes to their mind they feel free and express it. And at times, it makes the environment fun.*" *(Case 1 – Head of Engineering)*

On whether, the differences in communication skills between the various cohorts influence employment outcomes, the Head of Engineering added again that:

> "*Most people want someone who can get the work done, and so if you can get the work done it won't matter much.*"

The Senior Software Engineer also added:

> "*There are differences [in communication skills]. So a lot of Gen Zs prefer to use social media, they will prefer to have calls the same way we are having a call [virtual interaction] right now. But a lot of the older generation would prefer face to face meetings for everything. They sort of see the value in that, unlike the other [young] generations*" *(Case 1 – Senior Software Engineer)*

Responding on whether the differences in communication skills influence employment outcomes,the Senior Software Engineer added that:

> "*What I will say is that if you are a young Gen Z for instance, probably it is your first job. So it is understandable if your communication skills is not really good. But you might still get a pass, as time goes on you improve [on your communication skills]. But if you are coming from Gen Y, Gen X, then probably it might be quite important. So at least at that age, the expectation is that you already even have like 3 to 4,5, years of experience behind you, so you should be able to communicate effectively.*"

From the responses given, it is clear that the various generational cohorts exhibit different communication skills. Older generational cohorts prefer formal communication mediums such as emails whereas younger generations (Gen Y and Z) would prefer the use of social media for

interaction (Maan & Srivastava, 2023). However, these differences do not impact employment except for senior or management roles involving Gen X or Y. In summary, generational age moderates the relationship between communication skills and employment only with respect to older generations (Gen X and Y) or management positions.

5.4.9.3 Impact of Generational Age on Problem-Solving Ability and Employment

This section presents findings on whether there are differences in problem-solving abilities across the various generational cohorts (Gen X, Y, and Z) and how they affect the employment outcomes of software engineers.

The Head of Engineering said:

> "*Yes, so we are currently building a system, and you see the Y and Z, because they are exposed to a lot, they have shortcuts. Can't we try this thing? Let's try this thing. So their problem-solving skills, though it is arguable but some of them are very smart. From my knowledge, I think some of them are very smart. Yours is just, okay I have this, what is the best approach? Before you realize, they give you [a solution]. So they [Gen Y and Z] help in problem solving a lot.*" *(Case 1 – Head of Engineering)*

He added that:

> "*Yes, that is why a lot of people want younger people to do the development. Because, there is nothing occupying your mind [if you are younger]. You want fun and stuffs, so when the work is given, you should get it done.*"

The Program Manager also added:

> "*When it comes to the code itself, the newer generations tend to do a better job. Because they interact directly with the code.*" *(Case 1 – Program Manager)*

The Senior Software Engineer also added:

> "*So the Gen Zs are much more sharper. I have worked with other people who are way younger than me, and they are some of the best people I have ever worked with. The reason is a lot of the older generations especially Gen Y for instance, they are now starting out with life and maybe getting married, having kids, that kind of stuff. So they do not have time to upgrade themselves. But with the Gen Zs, it is quite different. They are probably still in school, under their parents, have enough time to learn new stuff.*" *(Case 1 – Senior Software Engineer)*

Given the above responses, it is evident that differences exist between the various generational cohorts with regard to problem-solving. As a result, software firms prefer Gen Z for their problem-solving ability, especially when it comes to coding or development. Younger generations are considered smarter and sharper when it comes to problem solving thereby enhancing their career outcomes. In summary, generational age moderates the relationship between problem-solving abilities and employment outcomes of software engineers.

5.4.10 Case two: Software Engineering Competencies (SEC) required by Employers in Ghana (SwF 2)

The software engineering competencies gathered from case two (SwF 2) are presented in the table 5.3. In the table, Co-founder / Project Manager is denoted as CoF, Head of Software Development as HSD, and Head of Mobile App Development as HMA.

Table 5.3 Software Engineering Competencies, Description and Importance (Case 2)

Technical Skills	Description	Reason of Importance
1. Knowledge in programming languages (Flutter, Kotlin, Java, JavaScript, PHP)	• "So the programming language is the code that you are going to write that is going to help you come up with that output that you seek." *(HSD)*	• "If you cannot write code or have knowledge in any aspect of development, you cannot be a software engineer." *(HMA)* • So the language is helping you come up with the output on the appropriate platform. *(HSD)*
2. Database management	• "So people are inputting their first name, last name, etc, and it's going to be stored somewhere. And so the database is what is keeping this information for you. *(HSD)* • "In terms of the different types of databases, we have MySQL, PostgreSQL, MongoDB, Oracle, Cassandra. *(CoF)*	• "most of the application you are going to be working on as an engineer, one way or the other you have to keep a certain information somewhere." *(HSD)* • "So if you have this database management skill, it's going to at least help you achieve that data storage aspect of the software development." *(HSD)*
3. DevOps	• "it is just about development and operations. If you combine them, that's what we mean by a DevOps engineer." *(CoF)* • "We call it elastic scalability, featuring AWS. So, this means that you are building the	• "So DevOps automation, it streamlines software engineering processes." *(CoF)* • "it helps in reliability of the software application…. DevOps engineers ensure that there is consistent and reliable software releases in terms of updates." *(CoF)*

	application in such a way that it can contain many users, be it 1 million, 2 million, the software should be able to contain it." *(CoF)*	• "not only that, but basically it helps you get an application from the development phase into the production phase." *(HSD)* • "reduces the downtime of the software application." *(CoF)*
4. Security	• "all the activities you perform during and after the development of the application to ensure that at least its vulnerability is lessened." *(HSD)* • "they also analyze risk assessment, look at the threat modeling, potential attack vectors, and scenarios." *(CoF)*	• "the extent to which you can lose resources in terms of funds when you are done with your application, and then you leave it out there" *(HSD)* • "Ensuring very well that there is confidentiality, there is integrity, there is availability of data on the system." *(CoF)*
5. Graphic design	• "So, basically, they deal with graphics. It can be motion graphics. So, in terms of UI, visual communication, in terms of brand and identity." *(CoF)*	• "in terms of brand communication to the public, graphic designers are very important." *(CoF)* • "sometimes it is the brand that communicates to the people" *(CoF)*
6. Research and innovation	• "You must investigate. You must get that data, analyze data. Then the data has to answer the questions that is being asked." *(CoF)* • "One of the biggest problems that we have in Africa is the industry	• "They came back and told us that the mobile app you have built is nice, but people are not downloading. .... our research team was able to do their research. They brought innovative ideas that we should give customers on the mobile app a scorecard, whereby if you use certain digital products,

	doesn't do research. Today if somebody comes for us to build a mobile app for the fellow, we need to research about the market." *(CoF)*	you are going to get a percentage." *(CoF)* • "So we do the research first. We are going to put innovative things, that Uber and Yango don't have." *(CoF)*
7. IT auditors (IT auditing skills)	• "these are professionals or technical people who evaluate an organization's information security system, the processes, and controls. They do this to ensure that the system is secured." *(CoF)*	• "They make sure that the software or application is efficient and effective for use." *(CoF)* • "It's important in terms of risk assessment. They are able to identify potential vulnerabilities and threats to IT system and data." *(CoF)*
8. Lead implementers (lead implementation skills)	• "responsible for putting IT solutions into action." *(CoF)* • "they make sure that all the software engineering processes are being implemented" *(CoF)*	• "these people are very important and crucial because, one, it brings productivity. It brings security. It also brings compliance. It brings business continuity and scalability" *(CoF)* • "They ensure the reliability of the software and return on investment." *(CoF)*
9. Testers (testing skills)	• "they are responsible for evaluating the software application to ensure that it meets all the requirements and is free from any defects, any vulnerability or any problems." *(CoF)*	• "if you don't test your software, you are not able to understand if the software is robust in terms of quality." *(CoF)* • "So testing is important to reduce risk, to save time and resources, and enhance security" *(CoF)*
10. Ethical hacking skills	• "we have the black hackers, the white	• "they identify vulnerabilities and weakness in the system" *(CoF)*

	hackers. So, ethical hackers are normally white hackers who hack to protect, they hack legally." *(CoF)*	• "It is important for conducting pen [penetration] testing." *(CoF)*
11. Communication skills	• "there must be good communication between you and your manager, or you and the people you are working with" (*HSD*)	• "there are certain things you have to communicate to your team or stakeholders" *(HMA)* • "generally, if you cannot communicate, you cannot lead." *(CoF)* • "it is important to be able to communicate to prevent misunderstanding" *(CoF)*
12. Project management	• "you should be able to break this particular project down to smaller bits and give timelines so that you can follow that gradually." (*HSD*)	• "almost every project has a deadline, a time frame allocated to it. So if you can manage projects, or if you can at least work on a project from the beginning to the end in an orderly manner to avoid confusion, this becomes a very good skillset to have." (*HSD*)
13. Teamwork	• "you have to know how to at least live among people, among the other developers and then work together as a group" (*HSD*) • "You can also have a platform like Jira or Asana to be able to collaborate and coordinate your	• "There is no software that you see running very, very well without being organized by a group of teams." *(CoF)* • "once you don't have that particular soft skill [teamwork] you are going to have a lot of challenges where you are either far ahead or behind what the group is working on." (*HSD*)

	software engineering." *(CoF)*	
14. Problem solving	• "Whatever you are working on, you should know the goal of what you are doing." (*HSD*)	• "with that [problem-solving skills], you are sure that whatever you are producing at the end of the day, it's not going to be a waste of time or resources. It's actually going to be something that you would use, or come out to be something of importance" (*HSD*)
15. Adaptability	• "The software engineering industry is evolving every time." *(HMA)*	• "Every now and then, there's a new software, there's a new AI model available for you to do your job better. So, if you are able to adapt and learn quickly and then implement these new things that have been introduced, I think in the long run, you will help the company and yourself as well." *(HMA)*
16. Analytical skills	• "software engineering is a field that needs a lot of concentration, analyzing data, analyzing the future, analyzing the past." *(CoF)*	• "You should be able to analyze data. You should be able to do system analysis and design. You should be able to debug codes." *(CoF)*
17. Emotional intelligence	• "you should be able to manage emotional things" *(CoF)*	• "when a customer calls you that, look, our servers are down. You don't panic in trying to solve the problem, and this is happening to a lot of amateur software developers because they don't understand the industry." *(CoF)*

Source: Author's construct based on findings

5.4.11 Achievement Orientation and Employment

The Head of Software Development said:

> "*A company like ours that focuses on developing software for clients, that should be one of the things that we should consider very important. And it is one thing that we take very seriously over here. To make sure there is customer satisfaction at the end of the day, then this must be something we cannot ignore.*" *(Case 2 – Head of Software Development)*

He added that:

> "*Yes, it does [affect employment outcomes]. Actually, for most of the developers that we get, whenever we decide that your skillset is good enough and want to enroll you, there will be a probation period, and this is something we look out for. Because we need to be sure that that self-motivation is there. We don't always have to be on you to complete tasks. So these are going to drive you to have a certain passion for whatever you are doing. And so with that self-motivation and tenacity, you will be able to deliver with less supervision.*"

The Head of Mobile Application also added:

> "*It does affect [employment outcomes] because, basically, if someone works here and then he comes and then he does not have the belief in what he is doing, it makes the team members work even harder. Because at the end of the day, I can come to the person and then say, please the code that you have written does not make sense. Then, instead of trying to make it understandable or prove that it is working based on your*

experience, you will gladly accept that it is wrong and then blame yourself for it. It does not help. It will set us back." *(Case 2 – Head of Mobile Application)*

The Co-founder/Project Manager also said:

"*So achievement orientation is one of the things that from our side we do every now and then because we want our developers to be very ambitious.*" *(Case 2 – Co-founder/Project Manager)*

He added that:

"*Yes, drive for excellence, self-motivation, tenacity, confidence in oneself, it's a critical thing in our recruitment process. But an organization cannot thrive where you don't have people thinking like this.*"

Hence, the responses from the participants clearly indicate that achievement orientation, which encompasses self-confidence, tenacity, self-evaluation, and drive, is important as it makes the work of other team members easier and helps ensure client satisfaction. Although this competency dimension is usually not assessed by software firms during their hiring processes, the responses indicate that it is evaluated during the probation period, which then influences the employment outcomes of software engineers.

5.4.12 Domain Knowledge and Employment

The Head of Software Development said:

"*Within our organization, it is only important for certain positions. When it comes to junior developers, then this is not something we are strict on. However, if you are going*

to play a leading role or a higher role, then we expect you to have a little bit of this [domain knowledge]." *(Case 2 – Head of Software Development)*

He added that:

"*So I think the answer to that will be yes and no but depends more on what we are employing for as a software engineer. We have in-house products, so if we think that you are going to play a leading role in a particular in-house product, and we think that the domain knowledge will be very key, then it becomes important to us.*"

The Co-founder/Project Manager also had this to say:

"*Those days no, but now yes. In the past, we would recruit young people directly from the university. Even if they were not fully competent, we would train them to fit into our system. Yes, we provided training so they could adapt to our way of working. But now, we no longer do that. For instance, we trained some people to pass ISO 27000 exams, including one who came from Canada, and they earned certifications. But in the end, they left the company.*" *(Case 2 – Co-founder/Project Manager)*

He added that:

"*So now we don't come and train you. If you are not competent [in the respective domain], you are gone. So, that is our philosophy now. We will not take such risks.*"

The above responses suggest that software firms consider domain knowledge as an important competency especially when an engineer is required to play a leading role in a particular in-house product. However, the responses indicate that whereas software firms are strict on domain knowledge when it comes to lead developer positions, it may be relaxed for junior

developer roles. In summary, domain knowledge is important and does influence the employment outcomes of software engineers.

5.4.13 Communication Skills and Employment

The Co-founder/Project Manager had this to say:

> "*As a software engineer, communication skills is not you going to the public to be talking like a politician or a radio presenter. It's you being able to communicate with your team about timelines, about delays, about a lot of stuff in software engineering.*" *(Case 2 – Co-founder/Project Manager)*

He added that:

> "*If you push your code to the test environment without telling me. If I send an email and you don't reply, it takes you three days to reply. If I send you a WhatsApp message and it takes you one week to reply in our group, it means that you are not reliable. So, you are not productive as well.*"

The Head of Software Development also said:

> "*Yes, it is very important. So communication is something that we value, or one of the competencies that we consider. In this organization, virtually all our work succeed when there is positive or better communication. So yes, this is necessary.*" *(Case 2 – Head of Software Development)*

He added that:

> "*Once there is broken or poor communication, what is going to happen is that there is going to be communication gaps, lapses in information. It is going to affect the end*

product. The timelines are going to be affected simply because certain information that should have come from you earlier on was delayed, or something that should get to somebody did not get there, and therefore there is miscommunication or no communication at all. Once this happens, there is going to be time loss, resource loss, and other things."

The Head of Mobile Application also added:

"*So during recruitment, one of the basic things that my organization looks for is communication skills. Because we are not working as a one-man company as some people do. We work with a team and being able to talk to others and understand what they mean, and when they need information is very important. So you not being able to communicate will affect your employment opportunity here.*" *(Case 2 – Head of Mobile Application)*

From the responses given, the communication skills of software engineers help software firms meet timelines and avoid delays. In addition, software engineers who are able to give timely feedback and reports on work done create an impression of being reliable and productive. Hence, since software development is a team effort, being able to communicate effectively is important and thus influences the employment outcomes of software engineers.

5.4.14 Problem Solving Ability and Employment

Concerning problem-solving ability and its influence on employment, the Head of Software Development said:

"*So virtually everything you are going to be working on as an engineer is because you want to solve a problem. You want to come out with a solution. And so if at the end of*

the day, you are going to come and do your coding, we hope that at the end of the day whatever you are coding can be used by somebody to address a problem." *(Case 2 – Head of Software Development)*

He added that:

"*We do not expect that you are always waiting for somebody to come and spoon-feed you with how to solve a problem. If that happens, the software you are going to develop is going to be so rigid, because you have not thought of all the possible scenarios. So it is something we look at from the beginning.*"

The Head of Mobile Application also added:

"*Being able to solve problems right away, sometimes with least supervision is very important because it is not every day that we are all here. Obviously, that is the basic thing software engineers do. They solve problems. So, if you cannot solve problems, then I think it will really affect your employment opportunity here.*" *(Case 2 – Head of Mobile Application)*

The Co-founder/Project Manager also had this to say:

"*So it is one of the things we look out for when we are recruiting new software engineers in our company. And we had a meeting today in the office, and it is one of the things that we are considering. If you cannot do this [problem solving], you are out.*" *(Case 2 – Co-founder/Project Manager)*

From the responses given, it is clear that problem-solving ability is one of the basic requirements of a software engineer. It helps the software engineer to come up with solutions

with the least supervision that will solve the software needs of clients. In summary, problem-solving ability is important to software firms and does affect the employment outcomes of software engineers.

5.4.15 Professional Work Ethic Ability and Employment

Responding to professional work ethic ability and its effect on employment outcomes, the Head of Software Development said:

> "*It is one of the competencies that we consider. The software development space actually comes with a lot of self-responsibility. You have to be responsible, you have to ensure that professionalism is part of you, especially considering our company where we are dealing with clients, external clients, not only our in-house products. So with this, we expect that when a client's information is available to you, you don't expose that.*" *(Case 2 – Head of Software Development)*

He added that:

> "*We expect that when the client pays this job should be done, and that this is what I want within this time frame, you are not lazy, you will put in the effort to make sure that we satisfy the client. So these ethics and these competencies are things that are important to us as a company.*"

The Head of Mobile Application also added:

> "*Within my organization, it is very important to be professional. When a task is assigned to you, you have to be able to deal with it and move on.*" *(Case 2 – Head of Mobile Application)*

The Co-founder/Project Manager also added:

> "*It's very important. No matter how good you are, if you are not disciplined, you cannot work with us.*" *(Case 2 – Co-founder/Project Manager)*

From the responses given, software engineers who have professional work ethic ability are self-responsible, exhibit professionalism, are hardworking, and these go a long way to ensuring client satisfaction. In summary, professional work ethic ability is important for software firms and does affect the employment outcomes of software engineers.

5.4.16 Teamwork Ability and Employment

As to whether teamwork ability is important and does influence employment outcomes of software engineers, the Head of Software Development said:

> "*Our work can be considered successful if the team is working and progressing well. Because software engineering, as you know has different aspects. The designers are there, the programming itself is there, I mentioned security and all that. Teamwork means that you are able to understand a bit of the other person's work. You understand how your work will impact the other person's work, etc.*" *(Case 2 – Head of Software Development)*

He added that:

> "*For this one, I think not so much [influence on employment]. And I will say that because should you come in-house and we realize that you are not really a team player, then we just find projects, research works, or things that you can handle by yourself that you don't need to depend on everybody. So we will not say that because you are not a good team player, we think that you are not qualified to work with us. You will be*

able to still work with us. We will find what suits you and gradually move you from that weakness."

The Co-founder/Project Manager also added:

"*If you are not a team player, all we just have to do is pull you in and mentor you to become a team player. We have software engineers in our organization, they are not team players, but they are good.*"

Hence, it is clear from the responses that software firms consider teamwork ability as an important competence. It helps software engineers understand the work of other team members, how their work will impact others, and how they all work together to achieve project goals (Raibulet & Fontana, 2018). However, from the responses, it is evident that teamwork ability is not used as a measure in disqualifying potential employees. In summary, teamwork ability although important, does not influence employment outcomes of software engineers.

5.4.17 Impact of Knowledge in Programming Languages

This section presents the findings of case 2 on the moderating effect of knowledge in programming languages on competencies and employment.

5.4.17.1 Impact of Knowledge in Programming Languages on Domain Knowledge and Employment.

The Head of Software Development said:

"*So in terms of the domain, we would not employ you and say we are looking for a software engineer. Probably we would streamline it a bit or we would reduce it to a particular field or domain. Like I said, it could be for mobile. So in that regard, we*

make sure that the languages that you can write are the ones that we use in writing applications for mobile because it is not one for all." *(Case 2 – Head of Software Development)*

He added that:

"*Like I said, we are recruiting you for a specific task or role or job. And so during the recruitment, we want to be sure that whatever we are employing you for, you are able to deliver in that specific task. So it affects employment in our company.*"

The Head of Mobile Application also added:

"*So the programming knowledge really enhances the domain knowledge of software engineers in a way that having knowledge in maybe let's say web development, that's HTML, CSS, JavaScript, usually enables the person to understand where or what we are trying to achieve in the long run. So if you have those kinds of programming knowledge, you are able to understand the domain that you are working in and able to produce the output that is expected of you.*" *(Case 2 – Head of Mobile Application)*

He added that:

"*Even with the programming knowledge that you have, you should be able to implement them in such a way that it will fit well with the domain. So that way you are able to guarantee your employment opportunity here in my organization.*"

From the responses, knowledge in programming languages helps software engineers to understand their respective domain domains better, be it in web or mobile application development. This goes a long way to affect the employment outcomes of software engineers.

In summary, knowledge in programming languages moderates the relationship between domain knowledge and employment outcomes of software engineers.

5.4.17.2 Impact of Knowledge in Programming Languages on Problem-Solving Ability and Employment

Here, the Head of Software Development said

> "*I would call this a tech stack and that is to say you being able to choose the right programming language and the tools to solve a problem. By that I mean it is not one for all. You cannot say that I know one language so I'm going to use it to write something for every problem that I will face. So based on the problem that you have, if we are trying to develop or solve a database problem with Microsoft, as an example, we expect that the programming knowledge that you have or the language that you are writing will be geared towards solving problems for the Microsoft environment.*" *(Case 2 – Head of Software Development)*

He added that:

> "*If you are going into software engineering and you know the tech stack that you work in, what you are good at, the programming knowledge that you know, you are able to then understand the problem, know the best language or best tech stack to use to solve it.*"

The Head of Mobile Application also added:

> "*But then how do you even know that there is a problem within this particular aspect of the application? You need to have programming knowledge in that field to know that this is the part where the problem is coming from. So that knowledge in programming*

languages is very important, it enhances problem-solving skills." *(Case 2 – Head of Mobile Application)*

From the responses given, when software engineers are proficient in their tech stack, it helps them to identify and solve problems within their domain, be it the web, mobile, or any other domain. In addition, the responses suggest that in software engineering, no single programming language can solve every problem. Therefore, proficiency in multiple languages allows software engineers to choose the most suitable one for each specific problem (Kim et al., 2018). In summary, knowledge in programming languages enhances the problem-solving ability of software engineers thereby affecting their employment outcomes.

5.4.18 Impact of Generational Age

This section presents the findings of case 2 on the moderating influence of generational age on SEC and employment.

5.4.18.1 Impact of Generational Age on Domain Knowledge and Employment

The Head of Software Development said:

"*The knowledge the older ones have seems to be different as compared to the Zs. The older ones do a little bit more analysis, research work, and stuff like that. They make sure they have more facts at their disposal before they implement the solution. They are a little bit careful with what I have seen.*" *(Case 2 – Head of Software Development)*

He went on to add that:

> "*From my experience, with the older ones, there are two stages to making sure that the thing is perfect. The younger ones, the Zs, they rather go for it and rather check to see if there are issues.*"

He went further and said:

> "*I think it depends on the industry you are going to work in. For example, I think that if you are going to work in the IT team of a bank, I think for that one it becomes important because they might need to know if your domain knowledge is up to a certain level to give you that job as compared to let's say a startup software development company. They might not really worry about the level of knowledge in the domain.*"

The Head of Mobile Application also added:

> "*Yes, there are different domain knowledge across the generations. Because when you talk about Gen X, they are more stable-oriented. They would like to have a detailed explanation of how the thing is going to work. With Generation Z, they would like to throw anything and then see what will happen, then come back to get the results and then filter it down, and go on. They [Gen Z] are more pragmatic than Generation Y and X.*" *(Case 2 – Head of Mobile Application)*

He went on to explain that:

> "*If I'm going to work for an insurance company, they would prefer Generation X, and Y rather than Generation Z because data is very important. You cannot go and then throw anything anywhere and wait for a response and then now you are going to look*

for the actual answer. The Generation X and Y are very thorough. They find the answer or they would try to look for the answer ahead before getting there."

He further added that:

"*With Generation Z, you usually find them in the startups. Normal startup companies where they allow innovations and then pragmatic people to come up with ideas. Those Generation Z are the kind of people who are able to throw anything out there, try to filter it out, and then come up with some new ways of doing stuff. So this affects employment in a way in one way or the other.*"

The Co-founder/Project Manager also had this to say:

"*Yes [there are differences in domain knowledge] and it does affect [employment outcomes]. Today when you ask us to employ somebody who is 50 years or 45 years, apart from managerial or operational positions, we usually wouldn't hire candidates in that age group*". *(Case 2 – Co-founder/Project Manager)*

Hence, the above statements clearly indicate that differences in domain knowledge exist among the various generational cohorts. Whereas the older generations (X and Y) who are considered 'stable-oriented' do more analysis and research work before tackling issues, Gen Zs are rather take a more pragmatic approach to solving issues. This makes Gen X and Y more suited for organizations where data is paramount such as banks and insurance companies while Gen Zs are more suited for startups where their pragmatic approach will be useful in developing innovative software products. In summary, differences in domain knowledge exist among the various generational cohorts and does affect the employment outcomes of software engineers.

5.4.18.2 Impact of Generational Age on Communication and Employment

Here, the Head of Software Development explained:

> "*Yes, they exhibit different communication skills. Some prefer using emails, others just prefer using WhatsApp and other quicker means of communication. I think the older generation, the X, moves toward the official communication channels, the emails, and stuff like that. But I think the younger generation, they prefers quicker ways of sending messages. If WhatsApp is available, why not?*" *(Case 2 – Head of Software Development)*

On whether these differences in communication skills affect employment outcomes, he said:

> "*No, I think not necessarily. So far as we can get information across, I think that is what matters.*"

The Head of Mobile Application also added:

> "*Yes. In my experience, the X and Y are very detail-oriented. So normally their communication is kind of detailed and precise and then even the channels that they use is quite professional and very formal. With Generation Z, their communication is should I say, 'fake' in a way that they would throw information in a disorderly manner. You have to sit down and analyze what they are trying to say to get what they mean.*" *(Case 2 – Head of Mobile Application)*

From the responses given, whereas older generations (Gen X and Y) usually use formal means such as emails to communicate, the younger Gen Zs prefer the use of informal means such as WhatsApp (McCourt, 2012). Also, Gen X and Y are usually detailed-oriented and communicate in an orderly manner whereas Gen Zs generally do not pay much attention to

detail. However, as indicated in the responses, these differences are not enough to influence the employment outcomes of software engineers. In summary, generational age does not moderate the relationship between communication and employment outcomes of software engineers.

5.4.18.3 Impact of Generational Age on Professional Work Ethic Ability and Employment

The Head of Software Development said:

> "*Yes, there is a difference in how they approach problem-solving. Some go a little bit deeper, worrying about every other thing, and that is the X group, from what I have seen. But Z, not as much as X. Because like I said, there are more tools now to quickly identify problems. So they [Gen Zs] rely on that. I think with our team we rely on go for it, we will detect if there is a problem. So there is a difference. (Case 2 – Head of Software Development)*

He further explained that:

> "*Yes, what we do is that if we know that whatever we are employing you for, we expect you to manage yourself, then we go with the X. Because then we are sure you are going to follow more of the procedural things that will be laid in place. If you are in the Z, the Y group, we know that you are in the younger generation so we put a little bit of supervision on you.*"

The Head of Mobile Application also added:

> "*With certain organizations like insurance companies and banks, it matters because they cannot bring you to do trial and error. But for startups, they are there for innovations. They are there for new ideas to come out. Even a mistake from a Gen Z*

would end up becoming a big opportunity for the campaign. Because during the trial and error, they might hop onto something that might be very valuable to the company without them knowing." *(Case 2 – Head of Mobile Application)*

From the responses given, Gen Z tends to rely more on trial and error, which sometimes leads to innovation - a trait desired by startup software firms. In contrast, older generations, such as X and Y, are more methodical in their problem-solving approach, making them preferable in organizations like banks, where trial and error are not welcomed. In summary, there are differences in problem-solving abilities across the generational cohorts and this influences the employment outcomes of software engineers.

5.5 Chapter Summary

This chapter presented qualitative findings on the software engineering competencies that are sought by employers in Ghana, the influence of competencies on employment and the moderating influence of knowledge in programming languages and generational age. The findings from the interviews with respondents who occupy lead roles produced a series of key themes. The insights of the interviewees were valuable in achieving the research objectives and thus provided a vivid understanding of the impact of competencies on employment outcomes. The following chapter will analyze and discuss these findings and relate them to the study's overall research questions and framework.

CHAPTER SIX

ANALYSIS AND DISCUSSION OF FINDINGS

6.1 Chapter Overview

In the previous chapter, the findings from the two qualitative case studies were presented. This chapter analyzes and discusses these findings in detail. The chapter begins by analyzing and discussing the software engineering competencies that were identified in the findings. This is followed by an in-depth analysis and discussion of the propositions framed for this study. It seeks to explicitly address the main themes that provide evidence for the propositions formulated in the study.

6.2 Software Engineering Competencies required by Employers in Ghana

This section analyzes and discusses the competencies required by software firms in Ghana. This was ascertained by asking respondents about the technical and soft skills required for working as a software engineer in their respective organizations. Aside from outlining the competencies, respondents also gave a brief description and reasons why the competencies mentioned were important.

Table 6.1 Software Engineering Competencies required by Employers in Ghana

Competencies	SwF 1	SwF 2	Supporting Reference
Technical Competencies			
Proficiency in the relevant programming language (C#, Flutter, React, Golang, PHP, Java, etc)	✓	✓	Assyne et al. (2022); Stamm (2023); Mangiza and Brown (2020); Kusumasari et al. (2018)
Version control (GitHub)	✓	✓	Assyne et al. (2022); Mangiza and Brown (2020);

Database Management	✓	✓	Calazans et al. (2017); Kusumasari et al. (2018)
Software Development Life Cycle	✓		
DevOps		✓	
Security		✓	Assyne et al. (2022);
Data structures and algorithms	✓		
Project management platforms such as Jira and Asana	✓	✓	Assyne et al. (2022);
Testing and integration	✓	✓	Mangiza and Brown (2020); Assyne et al. (2022);
Graphic design		✓	
Research and innovation		✓	
IT auditors (IT auditing skills)		✓	
Lead implementers (lead implementation skills)		✓	
Ethical hacking skills		✓	
Soft Competencies			
Communication skills	✓	✓	Assyne et al. (2022); Galster et al. (2022); Mangiza and Brown (2020);
Interpersonal skills	✓		Galster et al. (2022)
Mindset for growth and learning	✓		Mangiza and Brown (2020);

Problem solving	✓	✓	Mangiza and Brown (2020); Assyne et al. (2022); Galster et al. (2022)
Collaboration	✓		Galster et al. (2022); Assyne et al. (2022); Mangiza and Brown (2020);
Adaptability	✓	✓	Assyne et al. (2022);
Time management	✓		Kusumasari et al. (2018)
Leadership skills	✓		Assyne et al. (2022); Galster et al. (2022)
Being ethical	✓		Groeneveld et al. (2020)
Self-motivation	✓		Mangiza and Brown (2020);
Prior work experience	✓		Jebreen and Nabot (2021); Oguz and Oguz (2019)
Teamwork		✓	Galster et al. (2022); Assyne et al. (2022); Mangiza and Brown (2020)
Analytical skills		✓	Galster et al. (2022); Assyne et al. (2022);
Emotional intelligence		✓	

Source: Author's construct based on findings

The competencies that are required by software firms in Ghana are outlined in Table 6.1. In total, 14 technical skills and 14 soft skills were identified. The finding shows that software firms require a blend of both technical and soft skills. In addition, although the identified competencies largely aligned with existing literature, a number of competencies that are unique to this study were discovered. This includes DevOps, research and innovation, understanding of the SDLCs, and ethical hacking among others. While some of these uniquely identified

skills, such as DevOps, IT auditing, ethical hacking, and lead implementation, are typically considered specialized software engineering roles, insights from the study reveal that software firms may be upskilling their employees to take on such tasks, as they are considered essential in modern development. For example, graphic design skills were mentioned as important, as they aid in User Interface (UI) design, visual communication, and creating a brand and identity for clients. Furthermore, although software engineering has been perceived as overly technical, the balance between technical and soft competencies points to the increasing importance of non-technical abilities of software professionals.

6.3 Influence of Software Engineering Competencies on Employment

This section analyzes and discusses the impact of competencies on employment outcomes. This will help in answering research question 2.

6.3.1 Achievement Orientation

The level of influence of achievement orientation on the employment outcomes of software engineers is shown below.

Table 6.2 Achievement Orientation Factor item

Factor Items	SwF 1	SwF 2	Supporting reference
Handling pressure and stress	So in FinTech, most of us don't really sleep, so we sort of assess you based on how you can handle stress. And how determined you are to work under pressure.	So these are going to drive you to have a certain passion for whatever you are doing. And so with that self-motivation and tenacity, you will be able to	Kusumasari et al. (2018)

		deliver with less supervision.	
Individual and Organizational Success	You need self-confidence to learn. So if you are not learning you will be outdated. If new technologies are out because you are not always abreast with trending things, you will be left behind, and if care is not taken, you can lose your job.	But an organization cannot thrive where you don't have people thinking like this.	İhsan et al. (2015); Tenzer and Yang (2020)

Source: Author's construct based on findings

The Achievement orientation construct has been explained as a multifaceted competency that includes an individual's drive for excellence, tenacity, self-motivation, confidence in oneself, and self-evaluation. Thus, software engineers who have a drive for excellence, are self-motivated, and are confident in their abilities are likely to have positive employment outcomes (Liu et al., 2016). From the findings as shown in Table 6.2, software engineers who exhibit achievement orientation have the needed passion and are self-motivated which enables them to handle pressure and stress. This is very important since software firms operate under high-pressure environments and, hence require persons who are self-motivated and have tenacity.

In addition, software firms require persons who exhibit achievement orientation as this will lead to individual and organizational success. When software engineers possess achievement orientation, their self-confidence, motivation, and drive for excellence influence them to learn and stay abreast with the latest technological trends which enhance their employability thereby

leading to individual success (İhsan et al., 2015). Moreover, as highlighted by the respondents, software firms cannot thrive when their professionals are lacking in terms of achievement orientation. Hence, achievement orientation is critical for organizational success, given that software firms are held to high-performance standards by clients. As a result, possessing abilities such as self-confidence and a drive for excellence, among others, is essential for delivering software solutions that meet clients' specific requirements. The findings are consistent with those of Tenzer and Yang (2020), who found that employees with achievement orientation are more likely to engage in creative deviance, a behavior that is considered essential for fostering organizational innovation and success. Similarly, Utsch and Rauch (2000), observed a strong positive relationship between achievement orientation and organizational innovativeness, which is essential in achieving organizational success. In terms of SEC literature, although existing literature has not fully explored the achievement orientation construct, authors such as Calazans et al. (2017), Mangiza and Brown (2020), and Assyne et al. (2022), have identified achievement orientation components such as self-confidence, passion for the job, and a positive attitude as important for software firms. From the discussions above, achievement orientation is an important competency that is desired by software firms. Hence, considerable evidence exists that supports the proposition that:

> *Achievement orientation, characterized by tenacity, self-motivation, self-confidence, self-positioning, and self-evaluation, influences employment outcomes of software engineers.*

6.3.2 Domain Knowledge

The influence of domain knowledge on the employment outcomes of software engineers is shown below.

Table 6.3 Domain Knowledge Factor item

Factor Items	SwF 1	SwF 2	Supporting reference
Required for industry-specific solutions	Before you will be able to build a FinTech application, a financial system, you should know a little about finance. You should know your Debit and Credit.	if we think that you are going to play a leading role in a particular in-house product, and we think that the domain knowledge will be very key, then it becomes important to us.	de Campos et al. (2024); Chatterjee (2017)
Importance of staying updated with industry trends	How up-to-date am I with their new releases, events, and things that they are actually working on? How updated am I with that information? So we need to know because sometimes they improve on those things.	In the past, we would recruit young people directly from the university. Even if they were not fully competent, we would train them to fit into our system. Yes, we provided training so they could adapt to our way of working. But now, we no longer do that.	Yazdanian et al. (2021)

Source: Author's construct based on findings

From the responses, as indicated in Table 6.3, domain knowledge is required for industry-specific solutions. Hence, possessing both technical and business knowledge of a particular software field will enable an individual to better understand that field, which can lead to the design and development of more complex software solutions such as a FinTech application. This confirms the claims by de Campos et al. (2024) and Chatterjee (2017) that having domain knowledge provides industry-specific expertise, which is essential for developing industry-specific solutions.

In addition, the responses indicate that software firms have in-house products that require professionals with the necessary domain knowledge to deploy, maintain, and update them. Hence, possessing the requisite technical and business skills specific to the industry makes a software professional more competent in handling these in-house products. Moreover, possessing domain knowledge implies that a software engineer is up to date with the latest technological trends within that field (Yazdanian et al., 2021). This enhances job prospects or career progression, as software firms may be unwilling to provide the necessary training in the respective domain. Hence, domain knowledge is important and does affect the employment outcomes of software engineers. There is therefore evidence to support the proposition that:

> *Domain knowledge have an influence on the employment outcomes of software engineers.*

6.3.3 Communication skills

The influence of communication on the employment outcomes of software engineers is shown below.

Table 6.4 Communication Factor item

Factor Items	SwF 1	SwF 2	Supporting reference
Critical for teamwork	Your communication should be able to convey the information that you want to share with the recipient. So you should aim at making sure that whatever you're trying to tell the other person sits well with the person. The person understands what you are asking them to do or what you are requesting from them, and they are able to give you the appropriate feedback.	It's you being able to communicate with your team about timelines, about delays, about a lot of stuff in software engineering.	Boies et al. (2015)
Affects project success	So if you are the developer who is leading that particular project, you have to be able to effectively communicate between the sponsors and the rest of the engineering team.	Once there is broken or poor communication, what is going to happen is that there is going to be communication gaps, lapses in information. It is going to affect the end product. The	Wu et al. (2017)

		timelines are going to be affected	
Bridges technical and non-technical language	A lot of developers are really technical. They really find it hard to convert that technical speak into a more human friendly speak. So it's quite important for a lot of developers to be able to express themselves		Ye et al. (2016)

Source: Authors construct based on findings

Software firms often undertake complex software projects that require teamwork and collaboration among software engineers to ensure successful execution. From the findings, teamwork thrives when there is effective communication, ensuring a mutual understanding of what needs to be done and the timelines that are associated with a software project. This aligns with the study by Boies et al. (2015), which also found that leadership styles in communication are a significant factor affecting team outcomes.

In addition, it is essential for developers leading software projects to effectively communicate and relay vital information between sponsors and the engineering team. This will help prevent communication gaps and lapses in information, thereby contributing to project success. This aligns with the findings of Wu et al. (2017), which identified a positive relationship between project success and communication willingness. Moreover, as noted in the findings, many developers are highly technical and often struggle to translate technical language into more user-friendly terms (Ye et al., 2016). This highlights the importance of communication, as it enables software engineers to express themselves in simple language that can be understood

by relevant stakeholders. Hence, the above findings reveal that software firms consider communication an essential competency as it promotes teamwork, ensures project success, and helps translate technical language into user-friendly. This means there is enough evidence to support the proposition that:

> *Communication skills has an influence on the employment outcomes of software engineers.*

6.3.4 Problem-Solving Ability

The importance and influence of problem-solving ability on the employment outcomes of software engineers is shown below.

Table 6.5 Problem-Solving Ability Factor item

Factor Items	**SwF 1**	**SwF 2**	**Supporting reference**
A core engineering skill	Because in our industry, in as much as we are building cool and fantastic applications, the core of it is all about problem-solving.	virtually everything you are going to be working on as an engineer is because you want to solve a problem. You want to come out with a solution.	Van Aken and Berends (2018)
Independent Problem Solving	you should be able to think analytically. When given a task, you should be able to break them down into micro tasks.	Being able to solve problems right away, sometimes with least supervision is very important because it is not every day that we are all here.	Mangiza and Brown (2020)

Problem-solving ability as a critical hiring requirement	when we test somebody during a software engineering interview, that's usually what we're testing for	So, if you cannot solve problems, then I think it will really affect your employment opportunity here	Rabelo et al. (2022); Ahmed et al. (2013)

Source: Author's construct based on findings

This section sought to ascertain the extent to which software firms consider problem-solving ability as important and whether they affect employment outcomes. The analyzed data reveals that software firms consider problem-solving ability as the underlying core of their activities. Thus, software firms exist basically to solve software problems, hence the success or otherwise of these firms largely depends on how well their engineers solve those problems. This is highlighted by Van Aken and Berends (2018), who also identify problem-solving as one of the core skills required by software development firms. In addition, as highlighted by the software firms, the ability to solve problems independently with minimal supervision is considered important, as lead engineers may have other critical tasks to handle or may be unavailable to assist when needed (Mangiza & Brown, 2020). Moreover, problem-solving ability is assessed in-depth during recruitment. As indicated in the findings, problem-solving skills are thoroughly assessed during technical interviews where candidates are put to the test for their problem-solving skills. These skills are further validated in the course of the probation period, ensuring that new hires can think analytically and handle problems efficiently.

Hence, the above analysis clearly indicates that problem-solving abilities influence the employment outcomes of software engineers. This is consistent with existing SEC literature, which has emphasized problem-solving ability as an essential competency required by software firms (Assyne et al., 2022; Groeneveld et al., 2020; Mangiza & Brown, 2020;

Phaphuangwittayakul et al., 2018; Rabelo et al., 2022). From the above discussion, there is enough evidence to support the proposition that:

Problem-solving ability influences the employment outcomes of software engineers.

6.3.5 Professional Work Ethic Ability

The importance and influence of professional work ethic ability on the employment outcomes of software engineers is shown below.

Table 6.6 Communication Factor item

Factor Items	**SwF 1**	**SwF 2**	**Supporting reference**
Meeting timelines and deadlines	It's important, especially when we have to meet timelines. If you have been given a task that you are supposed to complete in four hours, and it has taken you three days because probably you are engaged with other clients.	We expect that when the client pays this job should be done, and that this is what I want within this time frame, you are not lazy, you will put in the effort to make sure that we satisfy the client.	Kuutila et al. (2020); Mangiza and Brown (2020)
Punctuality and discipline	So if you are obeying your closing time, then you should also be ready to come early	No matter how good you are, if you are not disciplined, you cannot work with us.	Galster et al. (2022)

Source: Author's construct based on findings

Software firms face stringent timelines in relation to the projects they have to execute. This makes professional work ethic ability a very important competency. Having professional work ethic ability implies that software engineers possess time management skills and a task oriented ability, elements that are very useful in meeting deadlines. From the findings, software firms expect that when a client places an order for a project, the assigned software engineers will put in their best efforts to executive the project within the stipulated timeframes. Thus, being lazy and procrastinating assigned tasks are traits that are despised in software firms. This finding is supported by Kuutila et al. (2020), who emphasize that the ability to meet deadlines is critical in software engineering. In addition, software firms expect their engineers to exhibit punctuality and discipline. As indicated in the findings, lateness and indiscipline are not countenanced as these will adversely affect project timelines and ultimately client satisfaction (Galster et al., 2022). Hence, professional work ability is important and influences employment outcomes, as it enables software engineers to meet deadlines, be punctual, and demonstrate discipline. From the above discussion, enough evidence exist to support the proposition that:

> *Professional work ethic ability influences the employment outcomes of software engineers.*

6.3.6 Teamwork Ability

The importance and influence of teamwork ability on the employment outcomes of software engineers is shown below.

Table 6.7 Teamwork Ability Factor item

Factor Items	SwF 1	SwF 2	Supporting reference
Important for project success	As the saying goes, if you go alone you cannot do so much, but if you go with other people, you can reach far. So teamwork is very key. One way or the other, you will rely on other people to be able to complete your day-to-day activities.	Our work can be considered successful if the team is working and progressing well. Because software engineering, as you know has different aspects.	Oh et al. (2021); Rabelo et al. (2022)

Source: Author's construct based on findings

Teamwork Ability is considered important for both software firms in ensuring the success of the software projects. As was mentioned in the findings, a software project such as a school management system can have 2 or 3 software engineers working on it. There could be a front-end developer, a backend developer, and a graphic designer all working on one software project. This means there is interdependency amongst software engineers, which calls for collaboration among engineers to ensure the successful execution of the project. The assertion that 'If you go alone you cannot do so much, but if you go with other people, you can reach far,' sums up and stresses the importance of working with other people for the realization of organizational goals. This view also underscores the fact that, while individual effort is important, it takes a combined team effort to ensure the success of a project. Oh et al. (2021) and Rabelo et al. (2022) similarly highlight the importance of teamwork, identifying it as a critical skill in software engineering. However, although respondents emphasized the

importance of teamwork, they were adamant that it has less influence on employment outcomes.

Respondents from both firms admitted that teamwork ability is usually not assessed during recruitment, as organizational structures are already in place to facilitate collaboration among engineers. This is consistent with the findings of Lindsjørn et al. (2016), who observed that the impact of teamwork on performance can range from small to significant, suggesting that teamwork may similarly have less influence on employment outcomes. In summary, teamwork ability is important for project success since there are various facets of software development that needs to be aligned. Hence, there is enough evidence to support the proposition that:

Teamwork ability has an influence on the employment outcomes of software engineers.

6.3.7 Impact of Knowledge in Programming Languages on Domain Knowledge and Employment

The moderating influence of knowledge in programming languages on domain knowledge and employment outcomes of software engineers are discussed below.

Table 6.8 Knowledge in Programming Languages and Domain Knowledge Factor item

Factor Items	SwF 1	SwF 2	Supporting reference
Programming proficiency aligns with domain needs	if you are proficient in your programming language, and you also have an understanding of your domain, like the FinTech that I'm in right	we make sure that the languages that you can write are the ones that we use in writing applications for mobile	de Campos et al. (2024)

	now, it helps you to build solutions that are fine-tuned to solve problems that are actually faced within the industry	because it is not one for all.	
Programming Proficiency Drives Domain Understanding	Domain knowledge is centered around programming languages, so it affects everything. That's the key. So when someone is good at programming languages, that person will be well versed in domain knowledge.	So if you have those kinds of programming knowledge, you are able to understand the domain that you are working in and able to produce the output that is expected of you.	Cheng (2018)

Source: Author's construct based on findings

The interaction of programming proficiency and domain knowledge goes a long way in determining how well software engineers are able to develop solutions to satisfy various industry needs. It is evident from the software firms that programming knowledge enhances an engineer's capacity to grasp and apply domain-specific insights effectively. This means software engineers leverage their foundation of programming knowledge to become professionally competent in their respective domains as emphasized by de Campos et al. (2024). This enables them to develop solutions that are not only technically sound, but tailored to the needs of the respective industry, whether it is in FinTech, mobile development, or any other specialized field. In addition, the findings indicate that having a strong grasp of programming knowledge facilitates a deeper understanding of a software domain as found by Cheng (2018). This understanding goes beyond coding; it deals with understanding the bigger picture of the industry, how various systems interact, and being able to appreciate the long-

term effect of their work. This helps software engineers to develop solutions that are business-oriented and serve the interests of the client.

In summary, programming proficiency and domain knowledge are linked in such a way that programming skills play the role of an enabler for one's deeper understanding of the domain. As such, persons who are proficient in programming languages are able to excel and solve problems in their respective domains by leveraging their proficiency in programming languages which sets them apart in a competitive job market. From the discussions above, there is enough evidence to support the proposition that:

> *Knowledge in programming languages has a moderating influence on the relationship between domain knowledge and employment outcomes in software engineering.*

6.3.8 Impact of Knowledge in Programming Languages on Problem-Solving Ability and Employment

The moderating effect of knowledge in programming languages on the problem-solving ability and employment outcomes of software engineers is shown below.

Table 6.9 Knowledge in Programming Languages and Problem-Solving Ability

Factor Items	SwF 1	SwF 2	Supporting reference
Knowledge of multiple languages helps in problem-solving	certain languages are built for certain problems, so as you learn more languages, it will help you just a bit over there.	I would call this a tech stack and that is to say you being able to choose the right programming language and the tools to solve a problem. By that I	Qian and Lehman (2017); Ntinda et al. (2021); Mangiza and Brown (2020)

		mean it is not one for all. You cannot say that I know one language so I'm going to use it to write something for every problem that I will face.	

Source: Author's construct based on findings

From the findings, programming languages are not one for as they are meant to solve different kinds of software problems. This means proficiency in multiple programming languages expands the problem-solving capability of a software engineer. Hence being proficient in only one language limits one's problem-solving ability, but being proficient in multiple languages allows the software engineer to use the best tool for particular challenges, which ensures effective problem-solving. In addition, proficiency in multiple languages brings more flexibility, thus widening an engineer's toolset which leads to better problem-solving. Thus, when software engineers are not limited to a single language, they can approach problems from various perspectives with a deeper range of techniques for finding a solution that works optimally. The ability to choose the proper language depending on the given problem, allows them to enhance problem-solving capabilities and provide more productivity in a wide range of domains and project requirements. Therefore, the findings of this study reinforce those of Qian and Lehman (2017), Ntinda et al. (2021), and Mangiza and Brown (2020), further affirming their findings on the importance of knowledge in programming languages in problem-solving.

In summary, software engineers enhance their career prospects when they are knowledgeable in multiple programming skills since their flexibility in employing different languages means they are adaptable to diverse tasks and problems, making them significant contributors in any

software development environment. Drawing on the above findings, there is evidence to support the proposition that:

> *Knowledge in programming languages has a moderating influence on the relationship between problem-solving ability and employment outcomes in the software engineering sector.*

6.3.9 Impact of Generational Age on Domain Knowledge and Employment

The moderating effect of generational age on domain knowledge and employment outcomes of software engineers are discussed below.

Table 6.10 Generational Age and Domain Knowledge Factor item

Factor Items	**SwF 1**	**SwF 2**	**Supporting reference**
Gen X: Deep knowledge but slower adaptation	The X [Gen X], it's like what they know, changing is difficult. They learned the thing the hard way... But the knowledge [domain knowledge] is the X, they know the thing.	The older ones do a little bit more analysis, research work, and stuff like that. They make sure they have more facts at their disposal before they implement the solution.	
Gen Y: Balanced knowledge and implementation	The Y [Gen Y], at least there is a much clearer understanding and there is little flexibility... The	Generation Y are more stable-oriented. They would like to have a detailed explanation of how the thing is going to work.	

	implementation is the Y.		
Gen Z: Tech-savvy and experimental	The Z [Gen Z], most of them have much knowledge [domain knowledge] because they are exposed to the thing at a very tender age, so they understand the concept better.... Gen Z is quite excelling at that [AI].	With Generation Z, they would like to throw anything and then see what will happen, then come back to get the results and then filter it down, and go on. They [Gen Z] are more pragmatic than Generation Y and X.	Mahmoud et al. (2021)

Source: Author's construct based on findings

From the findings, the various generational cohorts differ in terms of the depth and application of domain knowledge. As summarized in table 6.10, Gen X possess extensive domain knowledge founded on several years of experience. They are very analytical and cautious, always taking their time to do extensive research, gather many facts before providing solutions. Their knowledge and experience, particularly in traditional domains is unparalleled, but their approach is often slower and not adaptive to current technological trends. Because of this fact, Gen X professionals often end up in advisory or managerial positions where their years of experience and domain knowledge allow them to guide teams rather than participate directly in hands-on technical work.

Gen Y balances depth of knowledge of Gen X and the innovative, pragmatic approaches of Gen Z. Unlike Gen X, the findings revealed that they are more flexible, and adopt modern technologies, yet cautious in their approach to implementing solutions. Thus, they turn out to

be ideal employees in both strategic leadership and technical engagement roles. Very often, they are found in the management positions: CTOs, and project managers who are entrusted with the task of implementing the solutions by assiduously following business goals.

Moreover, from the findings, Gen Zs who are the youngest in the workplace are much more tech-savvy and knowledgeable in the latest technological trends such as artificial intelligence (AI). They are well known for their innovation and adaptability as established by Mahmoud et al. (2021) and Maan and Srivastava (2023). This makes Gen Zs more suitable for start-up firms where their knowledge in the latest tech trends is useful for innovation.

In summary, there are varying levels of domain knowledge across the different generational cohorts, a topic that has not been fully explored in existing literature. This makes some of the generations more suitable for certain positions than others. Whereas older generations, Gen X and Gen Y, have extensive domain knowledge and are more suitable for management roles, Gen Zs who are much more adaptable to modern technologies are much more suitable for technical/coding positions or in start-ups, where innovation is much needed. Hence, there is enough evidence to support the proposition that:

> *Generational age has a moderating influence on the relationship between domain knowledge and employment outcomes in the software engineering sector.*

6.3.10 Impact of Generational Age on Communication Skills and Employment

The moderating influence of generational age on communication and employment outcomes of software engineers is shown below.

Table 6.11 Generational Age and Communication Factor item

Factor Items	SwF 1	SwF 2	Supporting reference
Communication preferences across generations	They [Gen X] want things to be more formal; mostly emails, so they are mostly formal. For the Gen Y and Z, we like to use these platforms; Slack, Skype, WhatsApp, we want those things.	The older generation, the X, moves toward the official communication channels, the emails, and stuff like that… the younger generation prefers quicker ways like WhatsApp.	Maan and Srivastava (2023); Borg et al. (2023)
Formality vs. informality in communication	The Gen X are disciplined... But the Z, whatever comes to their mind they feel free and express it. And at times, it makes the environment fun.	With Gen X and Y, their communication is detailed and precise... With Generation Z, their communication is should I say, 'fake' in a way that they would throw information in a disorderly manner.	Maan and Srivastava (2023)
Do not impact employment outcomes	Most people want someone who can get the work done, and so if you can get the work done it won't matter much.	No, I think not necessarily [influence employment outcomes]. So far as we can get information across, I think that is what matters.	

Source: Author's construct based on findings

The findings indicate that generational age does influence workplace communication preferences, but the differences do not influence the employment outcomes of software engineers. Gen X were found to prefer formal communication mediums such as e-mail. This

generation upholds discipline and ensures clarity and order in their messages (Maan & Srivastava, 2023).

With Gen Y, they are accustomed to using both official and informal means of communication. Being the youngest generation in the workplace, Gen Z prefers informal and speedy communication, mostly through WhatsApp (Jimenez & Ford-Wilcox, 2022). Their casual attitude in terms of communication may create a relaxed and lively work atmosphere. These findings on communication skills of the various generational cohorts confirm that of Maan and Srivastava (2023) and Borg et al. (2023).

In spite of these generational differences in terms of communication skills, the findings indicate that employers are rather more concerned with whether the workers get their tasks done effectively, irrespective of the style or channel of communication utilized. Thus, as long as the communication achieves its purpose, be it with formal emails or informal WhatsApp chats, the particular communication style takes a back seat to the completion of the task at hand. As a result, there is insufficient evidence to support the proposition that:

> *Generational age has a moderating influence on the relationship between communication skills and employment outcomes in the software engineering sector.*

6.3.11 Moderating Influence of Generational Age on Problem-Solving Ability and Employment

The moderating effect of generational age on the Problem-Solving Ability and employment outcomes of software engineers is shown below.

Table 6.12 Generational Age and Problem Solving Ability Factor item

Factor Items	SwF 1	SwF 2	Supporting reference
Gen X: Methodical and thorough problem-solving	They [Gen X] are disciplined, methodical…. They consider various factors before finding a solution.	The X group goes a little bit deeper, worrying about every other thing.	
Gen Y: Balanced and flexible approach	The Y and Z, because they are exposed to a lot, they have shortcuts... some of them are very smart.		
Gen Z: Fast, experimental, and tech-driven	The Gen Zs are much more sharper... They have enough time to learn new stuff.	Because during the trial and error, they might hop onto something that might be very valuable to the company without them knowing	

Source: Author's construct based on findings

The findings indicate that the various generational cohorts approach problem-solving from different perspectives. Gen Xers are known to be very disciplined and methodical in that they consider various factors before they finally bring out a solution to a problem. Gen Y who are known to usually possess a broader knowledge of the domain, are considered smart just like Gen Zs since they know a lot of shortcuts in solving problems. The Gen Zs are considered sharper and are abreast with the latest technology as they have enough time to learn new stuff. Their pragmatic approach and reliance on trial and error in solving problems may lead to

mistakes which can even end up being a valuable opportunity for the firm. However, these findings remain unexplored in existing literature.

In summary, as mentioned by respondents, Gen Xers may be preferred in environments such as banks and insurance companies as their methodical approach will be useful in avoiding mistakes with critical data. Gen Zers are preferred in start-up firms where innovations are mostly needed and environments that prefer fast-paced and iterative developments. Hence, there is enough evidence to support the proposition that:

> *Generational age moderates the relationship between problem-solving ability and employment outcomes in the software engineering sector.*

6.4 Chapter Summary

This chapter provided an analysis of the findings gathered from the interviews conducted with two separate software companies. The chapter focused on the analysis of the software engineering competencies required by employers in Ghana and the influence of achievement orientation, domain knowledge, communication skills, problem-solving ability, professional work ethic ability, and teamwork ability on employment. The chapter also analyzed the moderating effect of knowledge in programming languages and generational age on the relationship between competencies and employment. The analysis offered insights into the influence of competencies on employment grounded on two cases from the software engineering sector.

CHAPTER SEVEN

SUMMARY, CONCLUSION AND RECOMMENDATIONS

7.1 Chapter Overview

In the previous chapter, the findings from the two qualitative case studies were analyzed in detail. This chapter gives a summary, conclusion, recommendations for future research as well as the limitations of the study.

7.2 Summary of the Research Process

This study intended to evaluate the impact of software engineering competencies on employment while also investigating the moderating effects of generational age differences and proficiency in programming languages. To achieve this goal, the study began in Chapter One by providing a background to the research, formulating the problem statement, which highlighted the research gaps that informed the study, as well as outlining the purpose of the study. The chapter also included clearly defined objectives and research questions, the significance of the study, and the thesis organization.

In Chapter Two of this study, relevant literature on software engineering competencies, generational age, and programming languages was reviewed. In addition, key concepts such as fundamental software engineering activities and software process models were covered. A review of the software engineering landscape in Ghana was also covered. The chapter concluded with an empirical review of existing literature on software engineering competencies, along with an analysis of research gaps and directions for future research.

Chapter Three provided the theoretical framework for the study, beginning with a review of relevant theories used in similar research, such as the Theory of Planned Behavior and the

Social Cognitive Career Theory. The selection and justification of the Human Capital Theory were also explained. The chapter further conceptualized a model based on the Human Capital Theory, with constructs drawn from the main software engineering competency domains proposed by Liu et al. (2016). This chapter also had the propositions that were framed for the study.

Chapter Four presented the exact methodologies that were used for the study. This included the research paradigm adopted for the study, the use of qualitative research design, the selection of multiple case study as a research method, the use of convenient and purposive sampling methods, and the selection of semi-structured interviews as a data collection method. The means of analyzing the collected data through thematic analysis were also explained.

In Chapter Five, the findings from the two case studies conducted were presented. This was useful in understanding the software engineering competencies that are sought by employers in Ghana, the influence of competencies on employment, and the moderating influence of knowledge in programming languages and generational age. The findings provided a vivid understanding of the impact of competencies on employment outcomes.

Chapter 6 synthesized the findings from Chapter 5 to form key themes for addressing the propositions framed in Chapter 3, with a focus on the impact of competencies on employment outcomes. The chapter analyzed the results through the conceptual model derived from Human Capital Theory. It also contextualized the findings within the literature reviewed in Chapter 2, the theoretical framework outlined in Chapter 3, and the insights from Chapter 5. This analysis enabled the development of several propositions grounded in the empirical findings.

Furthermore, the chapter laid the groundwork for a refined research framework as shown in Figure 7.1 that integrates the study's empirical findings.

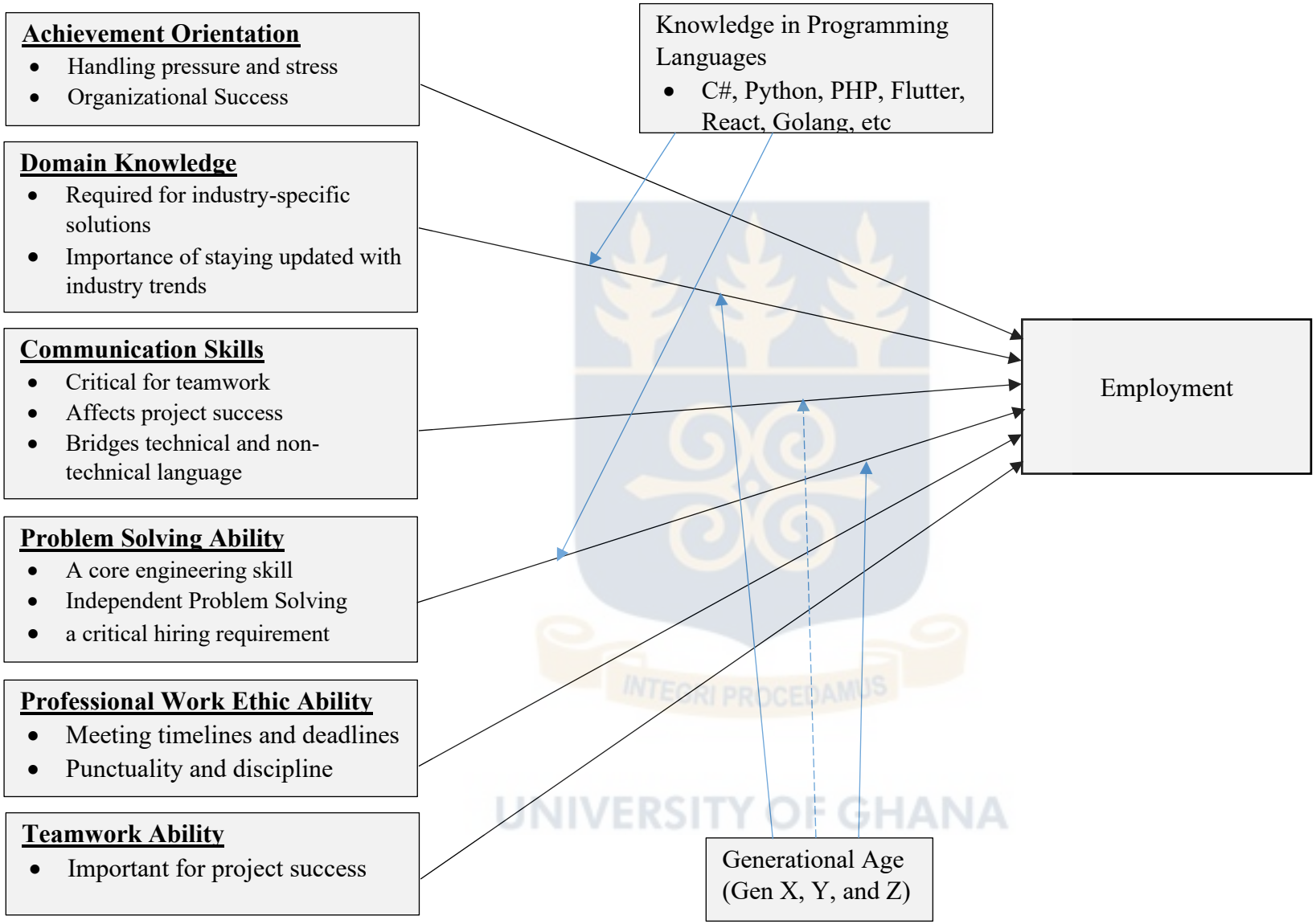

Figure 7.1 **Revised Conceptual Framework based on findings from Software Firms**

Table 7.1 Postulations made from findings

No	Propositions	SwF 1	SwF 2
1	Achievement orientation, characterized by tenacity, self-motivation, self-confidence, self-positioning, and self-evaluation, influences employment outcomes of software engineers.	✓	✓
2	Domain knowledge have an influence on the employment outcomes of software engineers.	✓	✓
3	Communication skills has an influence on the employment outcomes of software engineers.	✓	✓
4	Problem-solving ability influences the employment outcomes of software engineers.	✓	✓
5	Professional work ethic ability influences the employment outcomes of software engineers.	✓	✓
6	Teamwork ability has an influence on the employment outcomes of software engineers.	✓	✗
7	Knowledge in programming languages has a moderating influence on the relationship between domain knowledge and employment outcomes in software engineering.	✓	✓
8	Knowledge in programming languages has a moderating influence on the relationship between problem-solving ability and employment outcomes in the software engineering sector.	✗	✓
9	Generational age has a moderating influence on the relationship between domain knowledge and employment outcomes in the software engineering sector.	✓	✓
10	Generational age has a moderating influence on the relationship between communication skills and employment outcomes in the software engineering sector.	✗	✗
11	Generational age moderates the relationship between problem-solving ability and employment outcomes in the software engineering sector.	✓	✓

Source: Author's construct

7.3 Addressing the Research Objectives and Questions

A summary of key findings for each research objective is provided in Table 7.2.

Table 7.2 Mapping out Study Objectives with Findings, Literature, and Contributions

Research Objectives	Findings	Extant Literature	Contributions, Implications, And Recommendations
1. To examine software engineering competencies required by employers in Ghana	1. The study identified 14 technical skills and 14 soft skills that are required by employers in Ghana. The technical skills included proficiency in programming languages, database management, Version control, etc whereas the soft skills included problem-solving ability, communication, and adaptability among others. 2. The study discovered 9 competencies that have not been reported by extant literature. These include competencies such as DevOps, understanding of the SDLCs, emotional intelligence, etc.	Stamm (2023) in a mixed methods study identified 42 attributes sought by employers in new hires, with competencies such as teamwork, having strong initiative, and problem-solving among the highly ranked skills. Mangiza and Brown (2020), identified 43 technical and soft skills that are required by start-up firms. The reported competencies included problem-solving, teamwork, build new software, among others.	The study has added to the literature on software engineering competencies. This is very important due to the constant changes in technological trends in the SE field. The study has shed light on the technical competencies and soft competencies required by employers in Ghana. This will help prospective software engineers to develop those skills that are in-demand by the software industry. The study also addresses gaps in existing literature given that research on software engineering competencies is scarce in Africa.

	3. The findings show that software firms require a blend of both technical and soft skills.		
2. To explore the impact of software engineering competencies on employment in Ghana	1. Findings from the study reveal that achievement orientation significantly influences the employment outcomes of software engineers. 2. The findings reveal that domain knowledge significantly influences the employment outcomes of software engineers. 3. The findings also reveal that communication influences the employment outcomes of software engineers. 4. The findings reveal that problem-solving ability influences the employment outcomes of software engineers. 5. The findings reveal that professional work ethic ability influences the	Existing literature, such as Kusumasari et al. (2018), Assyne et al. (2022), Stamm (2023), and Mangiza and Brown (2020), emphasizes the importance of competencies like problem-solving, communication skills, and domain knowledge, among others. For instance, Mangiza and Brown (2020) found teamwork, communication, and character traits such as problem-solving, strong work ethic, and algorithmic thinking among others as the skill requirements for start-ups.	Existing literature covers the competencies that are considered important for software engineering roles. However, existing literature has not explored the influence of these competencies. This study addresses the gap in existing literature by exploring the influence of competencies on employment. This represents a novel contribution to the existing body of knowledge on software engineering competencies.

	employment outcomes of software engineers. 6. The findings reveal that teamwork ability influences the employment outcomes of software engineers.		
3. To investigate the moderating influence of generational age and knowledge in programming languages on software engineering competencies and employment.	1. The findings indicate that knowledge in programming languages moderates the relationship between domain knowledge and employment. 2. The findings reveal that knowledge in programming languages moderates the relationship between problem-solving ability and employment. 3. The findings indicate that generational age moderates the relationship between domain knowledge and employment. 4. The findings indicate that generational age does not moderate	Cheng (2018) found that proficiency in programming languages strengthens the domain knowledge. In addition, Maan and Srivastava (2023), and Borg et al. (2023) found differences in communication skills among the various generational cohorts.	The study addresses gaps in the existing literature by exploring the moderating effect of knowledge in programming languages on the relationship between SEC and employment. In addition, the study addresses gaps in existing literature by exploring the moderating influence of generational age on the relationship between SEC and employment. This serves as a novel contribution to the already existing literature on generational age.

	the relationship between communication and employment. 5. The findings indicate that generational age moderates the relationship between problem solving ability and employment.		

Source: Author's own construction

7.4 Summary of the Research Findings

The first objective of this study sought to explore software engineering competencies required by employers in Ghana. From the responses obtained, the study identified 14 technical skills and 14 soft skills that are required by employers in Ghana. These included competencies that have been reported by existing literature such as proficiency in programming languages, database management, Version control, problem-solving ability, and communication. In addition, the study identified 9 competencies that had not been previously emphasized in literature: Software Development Life Cycle, DevOps, data structures and algorithms, graphic design, research and innovation, IT auditing, implementation leadership, ethical hacking, and emotional intelligence.

The second objective of this study sought to explore the impact of software engineering competencies on employment. From the responses obtained, all the competencies: achievement orientation, domain knowledge, communication skills, problem-solving ability, professional work ethic ability, and teamwork ability, had an impact on the employment outcomes of software engineers. However, the findings reveal that communication skills and teamwork ability have less influence on the employment outcomes of software engineers.

The third objective of this study sought to explore the moderating influence of generational age and knowledge in programming languages on software engineering competencies and employment. From the responses obtained, knowledge in programming languages moderates the relationship between domain knowledge and employment, as well as problem-solving ability and employment. In addition, the responses indicate that generational age moderates the relationship between both domain knowledge and employment, as well as problem-solving

ability and employment. However, the findings suggest that generational age does not moderate the relationship between communication and employment.

7.5 Conclusion

The software industry has become one of the most promising sectors in the world as both private and public sectors of countries seek to adopt digitalization in order to achieve efficiency in their processes. Ghana is no exception and possesses a sizable market for software that can be exploited by indigenous software companies. However, since the software industry is heavily reliant on the skills and abilities of its professionals, this study explored the competencies that are in-demand by employers, their influence on employment, and the moderating influence of knowledge in programming languages and generational age. The findings indicate 14 technical skills which include proficiency in programming languages, database management, version control, security, and DevOps among others. Similarly, 14 soft competencies were identified which include problem-solving, communication skills, and teamwork, among others. In addition, it was found that competency dimensions such as achievement orientation, domain knowledge, communication skills, problem-solving ability, professional work ethic ability, and teamwork ability, impact the employment of software engineers. Moreover, the study found that software engineers who are proficient in programming languages enhance their employment outcomes, as they are able to understand their respective domains and effectively identify and solve problems. Furthermore, this study confirms that the various generational cohorts (Gen X, Y, and Z) exhibit differences in domain knowledge, and problem-solving abilities which eventually affect their employment outcomes. The study however found that although the various cohorts exhibit different communication skills, they do not influence employment outcomes. The findings of the study have several implications which will be covered by the subsequent section.

7.6 Implications of the Study

The findings of this study have significant implications for academic research, policy, and practice.

7.6.1 Implication to Research

In terms of implications to research, this study has added to the already existing body of knowledge on SEC. It expands the literature on the competencies valued by the software industry, with the discovery of new competencies adding to those already identified in previous research. In addition, the SEC research domain has been given little attention in developing countries such as Ghana. This study therefore fills the gap in literature and serves as a foundation for further exploration of the SEC domain in Ghana and other developing countries.

Moreover, although generational age has been extensively covered in the literature, few studies have explored its application in software engineering, particularly in relation to competency development. By examining the moderating effect of generational age on the relationship between competencies and employment, this study offers valuable insights into the various generational cohorts. This serves to extend the already existing body of knowledge on generational age. Furthermore, this study demonstrates the versatility of the Human Capital Theory. Although the theory has been employed in diverse research domains including economics, human resource management, and entrepreneurship, it has not been utilized in the SEC domain. Though not a traditional IS theory, Human Capital theory was effective in this study in helping to identify those investments software engineers make in themselves that enable them to achieve positive employment outcomes.

7.6.2 Implication to Practice

In terms of practice, this study will inform both prospective and existing software engineers of the competencies that are sought after by employers in Ghana. This will aid them in career planning and personal development as they become aware of the competencies valued in the software engineering industry. In addition, employers and human resource managers can utilize the identified skills as a reference point for vetting and evaluating applicants in order to guarantee the selection of candidates who possess the requisite competencies. This will lead to a highly competent workforce, which in turn will improve software quality and lead to client satisfaction. This will also aid employers and management of software firms in developing a framework for training and professional development programs. Moreover, the insights shed on generational age will aid software firms in understanding the differences that exist across the various generational cohorts in terms of domain knowledge, communication, and problem-solving. This will enable them to leverage and harness the strengths of each cohort in achieving the organization's goals.

7.6.3 Implication to Policy

In terms of policy, findings from the study will guide the development of curriculum and certification programs for SE professionals. Thus, the study's findings serve to inform educational bodies and curriculum developers of the competencies valued by employers, encouraging their incorporation into academic programs. This could be in the form of updating course offerings, revising learning objectives, and incorporating more hands-on activities into the academic work of prospective software engineers. This will help address the undesirable gap between academia and industry.

Additionally, although there have been several reports of age biases in the tech industry, this study provides empirical evidence on potential age biases regarding hiring in the software industry. The findings on generational age which shed light on potential age bias in recruitment offer insights for the Ministry of Employment and Labour Relations, as well as other relevant policymakers, to initiate policies that promote age-neutral recruitment practices.

7.7 Research Limitations

This study has some limitations. To begin with, the study used only two software firms in the case study. This number could have been expanded to ensure greater generalizability of findings. In addition, the study sampled individuals who occupy top positions such as co-founder and head of engineering. Incorporating software engineers of lower ranks could have yielded a more comprehensive understanding of the competencies they consider crucial for their employers, along with their perspectives on how their generational cohort impacts their competency development and career progression. Furthermore, the study relied solely on qualitative methods to explore the competencies that are sought after by employers in Ghana. A mixed methods approach could have aided in validating the findings with a larger sample size.

7.8 Future Research Directions

The study provides several research avenues that can be covered by future research. To begin with, the competencies that were identified in the study can be further investigated by quantitative methods to validate them. In addition, future research can explore the topic using employees of lower rank as respondents to further validate the findings of this study. Moreover, the issue of generational age and its impact on employment should be further explored in the software industry and other relevant industries where perceptions of age bias exist.

Furthermore, future studies could use the competencies identified in this study as a basis in investigating the disparity between what the Ghanaian academia teaches and the requirements of industry. Finally, the responses from respondents suggested a lack of highly skilled software professionals in Ghana. Hence, future research can focus on developing a framework that can be used by the Ghanaian software industry to attract, nurture, and retain highly qualified software professionals.

REFERENCES

Abdulkareem, S. A., & Abboud, A. J. (2021). Evaluating Python, C++, JavaScript and Java Programming Languages Based on Software Complexity Calculator (Halstead Metrics). *IOP Conference Series: Materials Science and Engineering*, *1076*(1), 012046. https://doi.org/10.1088/1757-899x/1076/1/012046

Acharya, A. S., Prakash, A., Saxena, P., & Nigam, A. (2013). Sampling: why and how of it? *Indian Journal of Medical Specialities*, *4*(2). https://doi.org/10.7713/ijms.2013.0032

Adel, A., & Abdullah, B. (2015). A Comparison Between Three SDLC Models Waterfall Model, Spiral Model, and Incremental/Iterative Model. *IJCSI International Journal of Computer Science Issues*, *12*(1), 106–111. https://www.academia.edu/10793943/A_Comparison_Between_Three_SDLC_Models_Waterfall_Model_Spiral_Model_and_Incremental_Iterative_Model

Adjaye, D. A. (2023). *A Glimpse at Generations and Ghana's Recent History*. https://www.linkedin.com/pulse/glimpse-generations-ghanas-recent-history-daniel-adjetey-adjaye/

Adom, K., & Asare-Yeboa, I. T. (2016). An evaluation of human capital theory and female entrepreneurship in sub Sahara Africa: some evidence from Ghana. *International Journal of Gender and Entrepreneurship*, *8*(4), 402–423.

Aggarwal, P. (2023). *Exploring Different Domains in Software: A Beginner's Guide*. https://medium.com/@pranshu1902/exploring-different-domains-in-software-a-beginners-guide-f5b67e1887a8

Ahiabenu, K. (2017, October 17). *Ghana's software industry - Take off or take flight*. https://www.ghanaweb.com/GhanaHomePage/features/Ghana-s-software-industry-Take-

off-or-take-flight-591642

Ahmed, F., Capretz, L. F., Bouktif, S., & Campbell, P. (2013). Soft skills and software development: A reflection from software industry. *International Journal of Information Processing and Management*, *4*(3), 171–191. https://doi.org/10.4156/ijipm.vol4.issue3.17

Ajzen, I. (1991). The Theory of Planned Behavior. *Organizational Behavior and Human Decision Processes*, *50*(2), 179–211.

Akdur, D. (2021). Skills Gaps in the Industry: Opinions of Embedded Software Practitioners . *ACM Trans. Embed. Comput. Syst.* , *20*(5). https://doi.org/10.1145/3463340

Akdur, D. (2022). Analysis of Software Engineering Skills Gap in the Industry . In *ACM Transactions on Computing Education* (Vol. 23, Issue 1). https://doi.org/10.1145/3567837

Akhtar, A., Bakhtawar, B., & Akhtar, S. (2022). Extreme Programming Vs Scrum: A Comparison Of Agile Models. *International Journal of Technology, Innovation and Management (IJTIM), 2(2), 80-96.*

Al-Msie'Deen, R., Blasi, A. H., & Alsuwaiket, M. A. (2021). Constructing a software requirements specification and design for electronic IT news magazine system. *International Journal of Advanced and Applied Sciences*, *8*(11), 104–118. https://doi.org/10.21833/IJAAS.2021.11.014

Alaidaros, H., Omar, M., & Romli, R. (2021). The state of the art of agile kanban method: challenges and opportunities. *Independent Journal of Management & Production, 12(8), 2535-2550.*

Ali, S., & Qayyum, S. (2021). A Pragmatic Comparison of Four Different Programming

Languages. *ScienceOpen Preprints*, *June*, 0–1. https://doi.org/10.14293/S2199-1006.1.SOR-.PP5RV1O.v1

Ali, S. S., Shoaib Zafar, M., & Saeed, M. T. (2020). Effort Estimation Problems in Software Maintenance - A Survey. *2020 3rd International Conference on Computing, Mathematics and Engineering Technologies: Idea to Innovation for Building the Knowledge Economy, ICoMET 2020*. https://doi.org/10.1109/iCoMET48670.2020.9073823

Allioui, H., & Mourdi, Y. (2023). Unleashing the Potential of AI: Investigating Cutting-Edge Technologies That Are Transforming Businesses. *International Journal of Computer Engineering and Data Science*, *3*(2), 2737–8543.

Alvi, M. (2016). *A manual for selecting sampling techniques in research*.

Amerson, R. (2011). Making a case for the case study method. *Journal of Nursing Education*, *50*(8), 427–428. https://doi.org/10.3928/01484834-20110719-01

Andrei, B. A., Casu-Pop, A. C., Gheorghe, S. C., & Boiangiu, C. A. (2019). A Study on Using Waterfall and Agile Methods in Software Project Management. *Journal of Information Systems & Operations Management*, 125–235.

Ankiilu, M. (2022). *Ghana's ICT Sector Could Reach $5 Billion by 2030 - African Eye Report*. https://africaneyereport.com/ghanas-ict-sector-could-reach-5-billion-by-2030/

Armitage, C. J., & Conner, M. (2001). Efficacy of the Theory of Planned Behaviour : A Meta-Analytic Review E Y cacy of the Theory of Planned Behaviour : A meta-analytic review. *The British Psychological Society*, *1*(1), 471–499.

Assyne, N., Ghanbari, H., & Pulkkinen, M. (2022). The essential competencies of software professionals : A unified competence framework. *Information and Software Technology*,

151(June), 107020. https://doi.org/10.1016/j.infsof.2022.107020

Awodiji, O. A. (2024). The Moderating Effect of Gender and School Type on the Nexus between Soft Skills and TVET Graduates' Employability. *Journal of Technical Education and Training*, *16*(1), 226–240. https://doi.org/10.30880/jtet.2024.16.01.016

Baxter, D., & Turner, N. (2021). Why Scrum works in new product development: the role of social capital in managing complexity. *Production Planning & Control, 1-13.*

Becker, G. S. (1962). Investment in Human Capital: A Theoretical Analysis. *Journal of Political Economy*, *70*(5, Part 2), 9–49. https://doi.org/10.1086/258724

Benbasat, I., Goldstein, D. K., & Mead, M. (1987). The Case Research Strategy in Studies of Information Systems. *MIS Quarterly*, *11*(3), 369–386. https://www.jstor.org/stable/248684?seq=1#page_scan_tab_contents

Bloomfield, J., & Fisher, M. J. (2019). Quantitative research design. *Journal of the Australasian Rehabilitation Nurses Association, 22(2), 27-30.*

Blundell, R., Dearden, L., Meghir, C., & Sianesi, B. (1999). Human Capital Investment: The Returns from Education and Training to the Individual, the Firm and the Economy. *Fiscal Studies*, *20*(1), 1–23. https://doi.org/10.1111/j.1475-5890.1999.tb00001.x

Boateng, R. (2016). *Research made easy*. CreateSpace Independent Publishing Platform.

Boies, K., Fiset, J., & Gill, H. (2015). Communication and trust are key: Unlocking the relationship between leadership and team performance and creativity. *Leadership Quarterly*, *26*(6), 1080–1094. https://doi.org/10.1016/j.leaqua.2015.07.007

Bontis, N., Seleim, A., & Ashour, A. (2007). Human capital and organizational performance: A study of Egyptian software companies. *Management Decision*, *45*(4), 789–801. https://doi.org/10.1108/00251740710746033

Borg, J., Scott-Young, C. M., & Borg, N. (2023). What Generation Z needs: the role of project-based organizations in creating career sustainability. *International Journal of Managing Projects in Business*, *16*(3), 571–591. https://doi.org/10.1108/IJMPB-12-2022-0273

Bourque, P., & Fairley, R. E. (2014). *Guide to the Software Engineering Body of Knowledge Version 3.0 (SWEBOK Guide V3.0)*. IEEE Computer Society.

Brown, E. (2019). *Web development with node and express: leveraging the JavaScript stack*. O'Reilly Media.

Brown, T. (2011). Are you a digital native or a digital immigrant? Being client centred in the digital era. *British Journal of Occupational Therapy*, *74*(7), 313. https://doi.org/10.4276/030802211X13099513660992

Calazans, A., Paldês, R., Masson, E., & Brito, I. S. (2017). *Software Requirements Analyst Profile : a descriptive study of Brazil and Mexico*. 204–212. https://doi.org/10.1109/RE.2017.22

Camus, A. (2021). *Popular Niche Programming Languages to Learn*. https://www.microverse.org/blog/popular-niche-programming-languages-to-learn-in-2020

Cavana, R. Y., Delahaye, B. A., & Sekaran, U. (2001). Applied Business Research: Qualitative and Quantitative Methods. *Brisbane: John Wiley and Sons Australia Pty. Ltd*.

Çera, G., Pagria, I., Khan, K. A., & Muaremi, L. (2020). Mobile banking usage and gamification: the moderating effect of generational cohorts. *Journal of Systems and Information Technology*, *22*(3), 243–263. https://doi.org/10.1108/JSIT-01-2020-0005

Chatterjee, J. (2017). Strategy, human capital investments, business-domain capabilities, and performance: a study in the global software services industry. *Strategic Management Journal*, *38*(3), 588–608.

Chen, J., Xia, X., Lo, D., Grundy, J., & Yang, X. (2021). Maintenance-related concerns for post-deployed Ethereum smart contract development: issues, techniques, and future challenges. *Empirical Software Engineering*, *26*(6). https://doi.org/10.1007/s10664-021-10018-0

Cheng, F. (2018). Languages, Frameworks, Libraries, and Tools. *Build Mobile Apps with Ionic 4 and Firebase: Hybrid Mobile App Developmen*, 29–66.

Chouhan, V. S., & Srivastava, S. (2014). Understanding Competencies and Competency Modeling — A Literature Survey. *IOSR Journal of Business and Management*, *16*(1), 14–22. https://doi.org/10.9790/487x-16111422

Ciriello, R. F., Richter, A., & Mathiassen, L. (2024). Emergence of creativity in IS development teams: A socio-technical systems perspective. *International Journal of Information Management*, *74*(June 2022), 102698. https://doi.org/10.1016/j.ijinfomgt.2023.102698

Cogin, J. (2012). Are generational differences in work values fact or fiction? Multi-country evidence and implications. *International Journal of Human Resource Management*, *23*(11), 2268–2294. https://doi.org/10.1080/09585192.2011.610967

Collier, A. (1994). *Critical realism: an introduction to Roy Bhaskar's philosophy*.

Conforto, E. C., Salum, F., Amaral, D. C., Da Silva, S. L., & De Almeida, L. F. M. (2014). Can Agile Project Management Be Adopted by Industries Other than Software Development? *Project Management Journal*, *45*(July), 21–34.

https://doi.org/10.1002/pmj

Cornacchione, E., & Daugherty, J. L. (2013). Trends in opportunity costs of U.S. postsecondary education: A national HRD and human capital theory analysis. *New Horizons in Adult Education and Human Resource Development*, *25*(2), 62–82. https://doi.org/10.1002/nha.20017

Creswell, J. . (2012). *Educational research: Planning, conducting, and evaluating quantitative and qualitative research*. Pearson Education, Inc.

Creswell, J. . (2013). Steps in Conducting a Scholarly Mixed Methods Study. *Steps in Conducting a Scholarly Mixed Methods Study*, 1–54. https://digitalcommons.unl.edu/cgi/viewcontent.cgi?article=1047&context=dberspeakers

Creswell, J. W. (2009). Research Design (Third Edit). *Retrieved from Https://Books.Google.Co.Za/Books?Hl=enandlr=lang_enandid=EbogAQAAQBAJandoi=fndan Dpg=PR1anddq=quantitative+creswellandots=cahMpXRAF8andsig=jN66MOd3r5nCuoPAbLJry9QwdQQ#v=onepageandq=quantitative Creswellandf=false*.

Daeboug, K. (2009). Human Capital and its Measurement. *OECD World Forum*.

Daneva, M., Wang, C., & Hoener, P. (2017). What the Job Market Wants from Requirements Engineers? An Empirical Analysis of Online Job Ads from the Netherlands . *2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)* , 448–453. https://doi.org/10.1109/ESEM.2017.60

Darke, P., Shanks, G., & Broadbent, M. (1998). Successfully completing case study. *Info Systems*, 273–289.

Darnoto, B. R. P., & Siahaan, D. (2021). MultiPhiLDA for Detection Irrelevant Software

Requirement Specification. *2021 International Conference on Computer Science, Information Technology, and Electrical Engineering, ICOMITEE 2021*, 92–97. https://doi.org/10.1109/ICOMITEE53461.2021.9650315

de Campos, V., David, J. M. N., Ströele, V., & Braga, R. (2024). Aligning technical knowledge to an industry domain in global software development: a systematic mapping. *Journal of Software: Evolution and Process, E2713*.

Deepa, V., Sujatha, R., & Baber, H. (2021). Ageing and Learning Agility –Mediating role of learning perception and Moderating role of technology leverage. *International Journal of Lifelong Education*, *40*(5–6), 514–531. https://doi.org/10.1080/02601370.2021.1991501

Dekhane, S., Xu, X., & Tsoi, M. Y. (2013). Mobile app development to increase student engagement and problem solving skills. *Journal of Information Systems Education*, *24*(4), 299–308.

Dice. (2018). *Ageism Still a "Major" Problem in Tech: Dice Survey*. https://www.dice.com/career-advice/ageism-tech-major-problem-survey

Doherty, E. (2023). *What is object-oriented programming? OOP explained in depth*. https://www.educative.io/blog/object-oriented-programming

Duncan, S. (2015). Test Driven Development, Refactoring and Pair Programming LiveLessons (Workshop). *Software Quality Professional, 17(3), 57.*

Ebert, C., & Hochstein, L. (2023). DevOps in Practice. *IEEE Software*, *40*(1), 29–36. https://doi.org/10.1109/MS.2022.3213285

Egerová, D., Komárková, L., & Kutlák, J. (2021). Generation Y and generation Z employment expectations: A generational cohort comparative study from two countries.

E a M: Ekonomie a Management, *24*(3), 93–109. https://doi.org/10.15240/TUL/001/2021-03-006

Estler, H. C., Nordio, M., Furia, C. A., Meyer, B., & Schneider, J. (2014). Agile vs. structured distributed software development: A case study. *Empirical Software Engineering*, *19*(5), 1197–1224. https://doi.org/10.1007/s10664-013-9271-y

Etikan, I. (2016). Comparison of Convenience Sampling and Purposive Sampling. *American Journal of Theoretical and Applied Statistics*, *5*(1), 1. https://doi.org/10.11648/j.ajtas.20160501.11

Fauzi, M. A., Tribiakto, H., Moniva, A., & Amir, F. (2023). *Systematic literature reviews on Rapid Application Development Information System Developments*. *4*(1), 57–64. https://doi.org/10.25008/bcsee.

Fisher, C. (2010). Researching And Writing A Dissertation, An Essential Guide For Business Students. *Essex: Pearson Education Limited.*

Fleming, J., Becker, K., & Newton, C. (2017). Factors for successful e-learning: does age matter? *Education and Training*, *59*(1), 76–89. https://doi.org/10.1108/ET-07-2015-0057

Forrier, A., De Cuyper, N., & Akkermans, J. (2018). The winner takes it all, the loser has to fall: Provoking the agency perspective in employability research. *Human Resource Management Journal*, *28*(4), 511–523. https://doi.org/10.1111/1748-8583.12206

Gabbrielli, M., & Martini, S. (2023). *Programming languages: principles and paradigms*. Springer Nature.

Gajria, N. (2022). *African developers: creating opportunities and building for the future*. https://blog.google/around-the-globe/google-africa/african-developers-creating-

opportunities-and-building-future/

Galster, M., Mitrovic, A., Malinen, S., & Holland, J. (2022). What Soft Skills Does the Software Industry *Really* Want? An Exploratory . In *Proceedings of the 16th ACM / IEEE International Symposium on Empirical* (pp. 272–282). Association for Computing Machinery . https://doi.org/10.1145/3544902.3546247

Garousi, V., Giray, G., Tuzun, E., Catal, C., & Felderer, M. (2020). Closing the Gap between Software Engineering Education and Industrial Needs. *IEEE Software*, *37*(2), 68–77. https://doi.org/10.1109/MS.2018.2880823

Gerring, J. (2004). What is a case study and what is it good for? *American Political Science Review*, *98*(2), 341–354. https://doi.org/10.1017/S0003055404001182

Ghaffarian, V. (2011). The new stream of socio-technical approach and main stream Information Systems research. *Procedia Computer Science*, *3*, 1499–1511. https://doi.org/10.1016/j.procs.2011.01.039

Gonçalves, L. (2018). Scrum: The methodology to become more agile. *Controlling & Management Review, 62(4), 40-42.*

Gregory, A. M., & Penela, D. (2023). Context-specific elective coursework and student's employability development: Application of social cognitive career theory in hospitality education. *Journal of Hospitality, Leisure, Sport and Tourism Education*, *33*(November), 100465. https://doi.org/10.1016/j.jhlste.2023.100465

Groeneveld, W., Jacobs, H., Vennekens, J., & Aerts, K. (2020). Non-Cognitive Abilities of Exceptional Software Engineers: A Delphi Study . In *Proceedings of the 51st ACM Technical Symposium on Computer Science* (pp. 1096–1102). Association for Computing Machinery . https://doi.org/10.1145/3328778.3366811

Groeneveld, W., Vennekens, J., & Aerts, K. (2021). Identifying Non-Technical Skill Gaps in Software Engineering Education: . *ACM Trans. Comput. Educ.* , *22*(1). https://doi.org/10.1145/3464431

Gullette, M. M. (2017). *Ending ageism, or how not to shoot old people*. Rutgers University Press.

Gupta, S., Xie, Y., & Zafar, H. (2024). Fintech and Digital Payments: Developing a Domain Knowledge Framework. *Journal of Information Systems Education*, *35*(2), 189–202. https://doi.org/10.62273/HIGA9274

Hasan, S., Yusof, R. J. R., Sofian, H., Ahmad, R., Jomhari, N., Sani, A. A., Ab Hamid, S. H., Norman, A. A., Obaidellah, U. H., Azzuhri, S. R., & Idris, M. Y. I. (2023). Preliminary Study for Identifying Personnel for Innovative Software Engineering Project. *IEEE Global Engineering Education Conference, EDUCON*, *2023-May*, 1–5. https://doi.org/10.1109/EDUCON54358.2023.10125110

Heričko, T., Brdnik, S., & Šumak, B. (2022). Commit Classification Into Maintenance Activities Using Aggregated Semantic Word Embeddings of Software Change Messages. *CEUR Workshop Proceedings*, *3237*, 1–12.

Herrity, K. (2023). *What Is Object-Oriented Programming (OOP)? A Complete Guide*. https://www.indeed.com/career-advice/career-development/what-is-object-oriented-programming

Hinai, M. R. Al, Bhuiyan, A. B., & Husin, N. A. (2020). the Moderating Effects of Gender, Career, Moral Mindset on the Relationship Between the Graduate Attributes and Readiness for Employability Among Engineering Colleges Graduates in Oman. *International Journal of Accounting & Finance Review*, *5*(3), 16–30. https://doi.org/10.46281/ijafr.v5i3.807

Hiranrat, C., & Harncharnchai, A. (2018). Using text mining to discover skills demanded in software development jobs in Thailand. *ACM International Conference Proceeding Series*, 112–116. https://doi.org/10.1145/3206129.3239426

Hoda, R. (2022). Socio-Technical Grounded Theory for Software Engineering. *IEEE Transactions on Software Engineering*, *48*(10), 3808–3832. https://doi.org/10.1109/TSE.2021.3106280

Hoover, A. (2024). *Ageism Haunts Some Tech Workers in the Race to Get Hired*. https://www.wired.com/story/ageism-haunts-tech-workers-layoffs-race-to-get-hired/

Ibezim, N. E., & Chibuogwu, C. I. (2017). *Computer Programming Competencies Required by Computer Education Graduates for Sustainable Employment*. *Rev. Eur.*

İHSAN, S., Ekici, S., Soyer, F., & Eskiler, E. (2015). Does self-confidence link to motivation? A study in field hockey athletes. *Journal of Human Sport and Exercise*, *10*(1), 24-35.

International Trade Administration. (2022). *Ghana - Information and Communications Technology (ICT)*. https://www.trade.gov/country-commercial-guides/ghana-information-and-communications-technology-ict

Jan, S. R., Shah, S. T. U., Johar, Z. U., Shah, Y., & Khan, F. (2016). An innovative approach to investigate various software testing techniques and strategies. *International Journal of Scientific Research in Science, Engineering and Technology (IJSRSET), Print ISSN, 23951990.*

Jebreen, I., & Nabot, A. (2021). An analysis of job advertisements for software engineering employability skills and knowledge: Jordan as a case study . In *Asia-Pacific Journal of Science and Technology* (Vol. 26, Issue 4). https://doi.org/10.14456/apst.2021.46

Jia, J., Chen, Z., & Du, X. (2017). Understanding Soft Skills Requirements for Mobile Applications Developers . *2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)* , *1*, 108–115. https://doi.org/10.1109/CSE-EUC.2017.29

Jimenez, S., & Ford-Wilcox, H. (2022). Generational differences in communication and translation to medical education. *Osteopathic Family Physician*, *14*(2), 31–34.

Johnson, C. L., Gutter, M., Xu, Y., Cho, S. H., & DeVaney, S. (2016). Perceived Value of College as an Investment in Human and Social Capital: Views of Generations X and Y. *Family and Consumer Sciences Research Journal*, *45*(2), 193–207. https://doi.org/10.1111/fcsr.12195

Johnson, S. E., & Hall, A. (2005). The prediction of safe lifting behavior: An application of the theory of planned behavior. *Journal of Safety Research*, *36*(1), 63–73. https://doi.org/10.1016/j.jsr.2004.12.004

Johnston, K. (2016). *Is the hot tech job market leaving its veterans behind?* https://www.bostonglobe.com/business/2016/03/05/tech-job-market-hot-but-older-workers-struggle/775HPU2OYc5i0Jhr3THTqM/story.html

Kabia, M. (2011). Contributions of Professional Certification and Information Technology Work Experience to Self-Reported Job Performance. *Business and Technology Management*, *PHD-BA*(January 2012). https://doi.org/10.13140/2.1.2260.4805

Kahkonen, A.-K. (2014). Conducting a Case Study in Supply Management. *Operations and Supply Chain Management: An International Journal*, *April*, 31–41. https://doi.org/10.31387/oscm090054

Kan, M. P. H., & Fabrigar, L. R. (2017). Theory of Planned Behavior. *Encyclopedia of*

Personality and Individual Differences, 1–8. https://doi.org/https://doi.org/10.1007/978-3-319-28099-8_1191-1

Kansal, J., Organisation, D., Satyawali, P. K., Ganju, A., & Organisation, D. (2012). Competency mapping in knowledge based organizations. *International Journal of Management*, *July 2018*.

Kao, K. Y., Hsu, H. H., Rogers, A., Lin, M. T., Lee, H. T., & Lian, R. (2022). I See My Future!: Linking Mentoring, Future Work Selves, Achievement Orientation to Job Search Behaviors. *Journal of Career Development*, *49*(1), 232–245. https://doi.org/10.1177/0894845320926571

Kaur, R., & Sengupta, J. (2013). *Software Process Models and Analysis on Failure of Software Development Projects*. *2*(2), 1–4. http://arxiv.org/abs/1306.1068

Kici, D., Malik, G., Cevik, M., Parikh, D., & Başar, A. (2021). A BERT-based transfer learning approach to text classification on software requirements specifications. *Proceedings of the Canadian Conference on Artificial Intelligence*, 1–13. https://doi.org/10.21428/594757db.a4880a62

Kim, M., Zimmermann, T., Deline, R., & Begel, A. (2018). Data scientists in software teams: State of the art and challenges. *IEEE Transactions on Software Engineering*, *44*(11), 1024–1038. https://doi.org/10.1109/TSE.2017.2754374

Kolog, E. A., Mensah, I., & Egala, S. B. (2024). Cyberloafing deterrence in the public sector of Ghana. *Electronic Journal of Information Systems in Developing Countries*, *August 2023*, 1–17. https://doi.org/10.1002/isd2.12316

Kropp, M., Meier, A., & Biddle, R. (2016). Teaching agile collaboration skills in the classroom. *Proceedings - 2016 IEEE 29th Conference on Software Engineering*

Education and Training, CSEEandT 2016, 118–127. https://doi.org/10.1109/CSEET.2016.27

Kuhn, T. S. (1970). The Structure of Scientific Revolutions. *The University of Chicago Press, Chicago*.

Kusumasari, T. F., Trilaksono, B. R., & Aisha, A. N. (2018). Software development team competencies to support software development project success . In *International Journal of Engineering and Technology(UAE)* (Vol. 7, Issue 4, pp. 156–161). https://doi.org/10.14419/ijet.v7i4.40.24424

Kuutila, M., Mäntylä, M., Farooq, U., & Claes, M. (2020). Time pressure in software engineering: A systematic review. *Information and Software Technology, 121, 106257*.

Laplante, P. A. (2007). What every engineer should know about software engineering. In *Engineer*. Taylor & Francis Group, LLC.

Law, P. (2010). Gaming outcome of accountants and human capital theory: Macau evidence. *Management Research Review*, *33*(12), 1174–1186. https://doi.org/10.1108/01409171011092211

Lebens, M., Finnegan, R., Sorsen, S., & Shah, J. (2021). Rise of the Citizen Developer. *Muma Business Review*, *5*(12), 101–111.

Lee, K. D. (2017). *Foundations of programming languages*. Springer.

Lehtinen, T. O. A., Mäntylä, M. V., Vanhanen, J., Itkonen, J., & Lassenius, C. (2014). Perceived causes of software project failures - An analysis of their relationships. *Information and Software Technology*, *56*(6), 623–643. https://doi.org/10.1016/j.infsof.2014.01.015

Lent, R. W., Brown, S. B., & Hackett, G. (1994). Toward a unifying scct and academic

interest, choice and performance. In *Journal of Vocational Behavior* (Vol. 45, pp. 79–122).

Leong, F. (2014). Positivist Paradigm. *Encyclopedia of Counseling, 2(Pat 2), 45523*.

Li, P. L., Ko, A. J., & Begel, A. (2020). What distinguishes great software engineers? *Empirical Software Engineering*, *25*(322–352).

Lindsjørn, Y., Sjøberg, D. I. K., Dingsøyr, T., Bergersen, G. R., & Dybå, T. (2016). Teamwork quality and project success in software development: A survey of agile development teams. *Journal of Systems and Software*, *122*, 274–286. https://doi.org/10.1016/j.jss.2016.09.028

Liu, D., Peng, W., & Liu, W. (2016). *Competency Evaluation Model for the Software Development Team*. *16*(Febm), 556–564. https://doi.org/10.2991/febm-16.2016.84

López-Alcarria, A., Olivares-Vicente, A., & Poza-Vilches, F. (2019). A systematic review of the use of Agile methodologies in education to foster sustainability competencies. *Sustainability (Switzerland)*, *11*(10), 1–29. https://doi.org/10.3390/su11102915

Lunsford, T. R., & Lunsford, B. R. (1995). The research sample, part I: sampling. *JPO: Journal of Prosthetics and Orthotics, 7(3), 17A.*

Lyons, D. (2016). *When It Comes to Age Bias, Tech Companies Don't Even Bother to Lie*. https://www.linkedin.com/pulse/when-comes-age-bias-tech-companies-dont-even-bother-lie-dan-lyons

Lyons, E. (2000). Qualitative data analysis: Data display model. *Research Methods in Psychology*, *2*, 269–280.

Maalej, W., Nayebi, M., & Ruhe, G. (2019). Data-Driven Requirements Engineering-An Update. *Proceedings - 2019 IEEE/ACM 41st International Conference on Software*

Engineering: Software Engineering in Practice, ICSE-SEIP 2019, 289–290. https://doi.org/10.1109/ICSE-SEIP.2019.00041

Maan, P., & Srivastava, D. K. (2023). Factors affecting team performance: An empirical study of Indian GenY and GenZ cohorts. *Equality, Diversity and Inclusion*, *42*(8), 986–1006. https://doi.org/10.1108/EDI-05-2022-0114

Magni, F., & Manzoni, B. (2020). Generational Differences in Workers' Expectations: Millennials Want More of the Same Things. *European Management Review*, *17*(4), 901–914. https://doi.org/10.1111/emre.12405

Mahmoud, A. B., Fuxman, L., Mohr, I., & Reisel, W. D. (2021). *" We aren ' t your reincarnation ! " workplace motivation across X , Y and Z generations*. *42*(1), 193–209. https://doi.org/10.1108/IJM-09-2019-0448

Mall, R. (2018). Fundamentals of software engineering. *PHI Learning Pvt. Ltd.*.

Mandal, A., & Pal, S. C. (2015). Identifying the Reasons for Software Project Failure and Some of their Proposed Remedial through BRIDGE Process Models. *International Journal of Computer Sciences and Engineering*, *3*(1).

Mangiza, P., & Brown, I. (2020). Requisite Skills Profile of Software Development Professionals for Startups. *ACM International Conference Proceeding Series*, 102–109. https://doi.org/10.1145/3410886.3410904

Marebane, S. M., & Hans, R. T. (2021). Software Engineering Ethics Competency Gap in Undergraduate Computing Qualifications within South African Universities of Technology. *International Journal of Advanced Computer Science and Applications*, *12*(4), 579–592. https://doi.org/10.14569/IJACSA.2021.0120474

Marimuthu, M., Arokiasamy, L., & Ismail, M. (2009). *Human Capital Development and Its*

Impact on Firm Performance : Evidence from Developmental Economics.

Matook, S., & Maruping, L. M. (2014). A competency model for customer representatives in Agile software development projects. *MIS Quarterly Executive*, *13*(2), 77–95.

McClelland, D. C. (1973). Testing for competence rather than for "intelligence". *The American Psychologist*, *28*(1), 1–14. https://doi.org/10.1037/h0034092

McCourt, D. M. (2012). The "Problem of Generations" Revisited: Karl Mannheim and the Sociology of Knowledge in International Relations. *Theory and Application of the "Generation" in International Relations and Politics*, 47–70. https://doi.org/10.1057/9781137011565_3

Meyer, N., Barr, E. T., Bird, C., & Zimmermann, T. (2019). The Daily Life of Software Developers. *IEEE Transactions on Software Engineering*, 1–18.

Mezmir, E. A. (2020). Qualitative Data Analysis: An Overview of Data Reduction, Data Display and Interpretation. *Research on Humanities and Social Sciences*, *10*(21), 15–27. https://doi.org/10.7176/rhss/10-21-02

Miles, M. B., & Huberman, A. M. (1994). Qualitative Data Analysis. In *Qualitative Data Analysis: An Expanded Sourcebook*. Thousand Oaks.

Mingers, J., Mutch, A., & Willcocks, L. (2013). Critical Realism In Information Systems Research. *MIS Quarterly, 37(3), 795-802*.

Mishra, A., & Alzoubi, Y. I. (2023). Structured software development versus agile software development: a comparative analysis. *International Journal of System Assurance Engineering and Management*, *14*(4), 1504–1522. https://doi.org/10.1007/s13198-023-01958-5

Mohammad, S. M. (2018). Streamlining DevOps automation for Cloud applications.

International Journal of Creative Research Thoughts, *6*(4), 2320–2882. www.ijcrt.orgwww.ijcrt.org

Monteiro, S., Almeida, L., & Aracil, A. G. (2016). Graduates' perceptions of competencies and preparation for labour market transition: The effect of gender and work experience during higher education. *Higher Education, Skills and Work-Based Learning*, *6*(2), 208–220. https://doi.org/10.1108/HESWBL-09-2015-0048

Moreira, A., Cesrio, F., Chambel, M., & Castanheira, F. (2018). Competences development and affective Commitment: mediation through employability and moderation by generation. *European Journal of Management Studies*, *23*(2), 123. https://doi.org/10.5455/ejms/289312/2018

Moustroufas, E., Stamelos, I., & Angelis, L. (2015). Competency profiling for software engineers: Literature review and a new model. *ACM International Conference Proceeding Series*, *01-03-Octo*, 235–240. https://doi.org/10.1145/2801948.2801960

Moy, J. H., Van Dyne, A., & Hattrup, K. (2023). An Investigation of the Moderating Effects of National Culture Values on the Interaction Between Job Insecurity and Employability on Employee Outcomes. *Journal of Cross-Cultural Psychology*, *54*(1), 114–141. https://doi.org/10.1177/00220221221119720

Mubarik, M. S., Chandran, V. G. R., & Devadason, E. S. (2017). Measuring Human Capital in Small and Medium Manufacturing Enterprises : What Matters ? *Social Indicators Research*, *137*, 605–623. https://doi.org/10.1007/s11205-017-1601-9

Mumford, M. D., Todd, E. M., Higgs, C., & McIntosh, T. (2017). Cognitive skills and leadership performance: The nine critical skills. *The Leadership Quarterly*, *28(1)*, 24–39.

Myers, M. D., & Avison, D. (2002). Qualitative research in information systems. *A Reader: Sage*.

National Communications Authority. (2017). *Determining the Contribution of the ICT/ Telecommunication Sector to GDP in Ghana*. https://nca.org.gh/wp-content/uploads/2021/11/ICT-contribution-to-GDP-in-Ghana-Stakeholder-Feedback-on-the-study2.pdf

Nelson, S., & Duxbury, L. (2021). *Breaking the mold : Retention strategies for generations X and Y in a prototypical accounting firm*. 155–178. https://doi.org/10.1002/hrdq.21414

Nerdrum, L., & Erikson, T. (2001). Intellectual capital: A human capital perspective. *Journal of Intellectual Capital*, *2*(2), 127–135. https://doi.org/10.1108/14691930110385919

Nguyen, Q. M. (2020). *How business requirements communicate with technology in software development: case Integrify Oy*.

Niknafs, A., & Berry, D. (2017). *The impact of domain knowledge on the effectiveness of requirements engineering activities*. *22*(80–133).

Ntinda, M., Apiola, M., & Sutinen, E. (2021). Mind the gap: Aligning software engineering education and industry in Namibia. *2021 IST-Africa Conference, IST-Africa 2021*, 1–8.

Nugroho, S., Hadi, S., & Hakim, L. (2017). Comparative Analysis of Software Development Methods between Parallel, V-Shaped and Iterative. *International Journal of Computer Applications*, *169*(11), 7–11. https://doi.org/10.5120/ijca2017914605

Ogheneovo, E. E. (2014). On the Relationship between Software Complexity and Maintenance Costs. *Journal of Computer and Communications*, *02*(14), 1–16. https://doi.org/10.4236/jcc.2014.214001

Oguz, D., & Oguz, K. (2019). Perspectives on the Gap Between the Software Industry and

the Software Engineering Education . *IEEE Access* , *7*, 117527–117543. https://doi.org/10.1109/ACCESS.2019.2936660

Oh, J., Lee, H., & Zo, H. (2021). The effect of leadership and teamwork on ISD project success. *Journal of Computer Information Systems*.

Otterbach, S., & Sousa-Poza, A. (2016). Job insecurity, employability and health: an analysis for Germany across generations. *Applied Economics*, *48*(14), 1303–1316. https://doi.org/10.1080/00036846.2015.1100248

Özyurt, Ö. (2015). Examining the critical thinking dispositions and the problem solving skills of computer engineering students. *Eurasia Journal of Mathematics, Science and Technology Education*, *11*(2), 353–361. https://doi.org/10.12973/eurasia.2015.1342a

Papadopoulos, G. (2015). Moving from Traditional to Agile Software Development Methodologies Also on Large, Distributed Projects. *Procedia - Social and Behavioral Sciences*, *175*, 455–463. https://doi.org/10.1016/j.sbspro.2015.01.1223

Parry, E., & Urwin, P. (2011). Generational Differences in Work Values: A Review of Theory and Evidence. *International Journal of Management Reviews*, *13*(1), 79–96. https://doi.org/10.1111/j.1468-2370.2010.00285.x

Patomäki, H., & Wight, C. (2000). After postpositivism? The promises of critical realism. *International Studies Quarterly, 44(2), 213-237.*

Payscale. (2021). *Top Tech Companies Compared*. https://www.payscale.com/data-packages/top-tech-companies-compared

Peel, K. L. (2020). Beginner'S Guide To Applied Educational Research Using Thematic Analysis. *Practical Assessment, Research and Evaluation*, *25*(1), 1–16. https://doi.org/10.7275/ryr5-k983

Perkins, K. A., & Herring, G. (2021). Generational perspective and influence within student affairs workplaces. *New Directions for Student Services*, *2021*(175), 73–82. https://doi.org/10.1002/ss.20398

Phaphuangwittayakul, A., Saranwong, S., Panyakaew, S., Inkeaw, P., & Chaijaruwanich, J. (2018). Analysis Of Skill Demand In Thai Labor Market From Online Jobs Recruitments Websites . *2018 15th International Joint Conference on Computer Science and Software Engineering (JCSSE)* , 1–5. https://doi.org/10.1109/JCSSE.2018.8457393

Presti, L. A., Van der Heijden, B., & De Rosa, A. (2024). Organizational predictors of employability and the moderating impact of boundaryless career attitude: A multi-wave study among Italian employees. *Human Resource Management Journal*, *June 2023*, 1–19. https://doi.org/10.1111/1748-8583.12575

Qian, Y., & Lehman, J. (2017). Students' misconceptions and other difficulties in introductory programming: A literature review. *ACM Transactions on Computing Education*, *18*(1), 1–24. https://doi.org/10.1145/3077618

Rabelo, D., Lopes, A., Mendes, W., De Souza, C., Gama, K., Monteiro, D., & Pinto, G. (2022). The Role of Non-Technical Skills in the Software Development Market . In *ACM International Conference Proceeding Series* (pp. 31–40). https://doi.org/10.1145/3555228.3555254

Raibulet, C., & Fontana, F. A. (2018). Collaborative and teamwork software development in an undergraduate software engineering course. *Journal of Systems and Software*, *144*, 409–422. https://doi.org/10.1016/j.jss.2018.07.010

Ratajczak, J. (2020). *ANAGEMENT OF AGE-DIVERSE TEAMS AND THE TYPE OF ORGANIZATIONAL CULTURE*. *1*(1).

Razzak, T., Pahi, M. H., & Aziz, A. (2023). Political Skills and Perceived Employability: Exploring the Mediating Role of Career Self-efficacy. *Pakistan Journal of Humanities and Social Sciences*, *11*(3), 3696–3706. https://doi.org/10.52131/pjhss.2023.1103.0649

Rola, P., Kuchta, D., & Kopczyk, D. (2016). Conceptual model of working space for Agile (Scrum) project team. *Journal of Systems and Software, 118, 49-63.*

Safi, K. Al. (2019). Navigating generational differences in the workplace. *The Wiley Handbook of Global Workplace Learning*, 181–199. https://doi.org/10.1002/9781119227793.ch11

Saltz, J., & Heckman, R. (2020). Exploring which agile principles students internalize when using a Kanban process methodology. *Journal of Information Systems Education, 31(1), 51.*

Sapada, A. F. A., Modding, H. B., Gani, A., & Nujum, S. (2017). The Effect of Organizational Culture and Work Ethics on Job Satisfaction and Employees Performance. *The International Journal of Engineering and Science (IJES)*, *6*(12), 28–36. https://doi.org/10.9790/1813-0612042836

Sapkota, B., Kayestha, M., Tiwari, D. R., & Shrestha, P. (2024). Democratic Leadership and Job Satisfaction:Moderating Role of Age and Gender. *International Research Journal of MMC*, *5*(2), 91–103.

Sarker, I. H., Faruque, F., Hossen, U., & Rahman, A. (2015). A survey of software development process models in software engineering. *International Journal of Software Engineering and Its Applications*, *9*(11), 55–70. https://doi.org/10.14257/ijseia.2015.9.11.05

Saunders, M., Lewis, P., & Thornhill, A. (2003). Research Methods for Business Students.

Harlow, England: Prentice Hall.

Schefer-Wenzl, S., & Miladinovic, I. (2019). Developing Complex Problem-Solving Skills: An Engineering Perspective. *International Journal of Advanced Corporate Learning (IJAC)*, *12*(3), 82. https://doi.org/10.3991/ijac.v12i3.11067

Sedelmaier, Y., & Landes, D. (2014). Software engineering body of skills (SWEBOS). *IEEE Global Engineering Education Conference, EDUCON*, *April 2014*, 395–401. https://doi.org/10.1109/EDUCON.2014.6826125

Sedelmaier, Y., & Landes, D. (2015). SWEBOS – The Software Engineering Body of Skills. *International Journal of Engineering Pedagogy (IJEP)*, *5*(1), 20. https://doi.org/10.3991/ijep.v5i1.4047

Semerikov, S., Striuk, A., Striuk, L., Striuk, M., & Shalatska, H. (2020). Sustainability in Software Engineering Education: A case of general professional competencies. *E3S Web of Conferences*, *166*. https://doi.org/10.1051/e3sconf/202016610036

Sharma, G. (2017). Pros and cons of different sampling techniques. *International Journal of Applied Research, 3(7), 749-752.*

Smith, C., Halinski, M., Gover, L., & Duxbury, L. (2019). Generational Differences in the Importance, Availability, and Influence of Work Values: A Public Service Perspective. *Canadian Journal of Administrative Sciences*, *36*(2), 177–192. https://doi.org/10.1002/cjas.1485

Smuts, S., & Smuts, H. (2022). Society 5.0 and the future of work skills for software engineers and developers . In *EPiC Series in Computing* (Vol. 84, pp. 169–182). https://doi.org/10.29007/9kzd

Sohaib, O., Solanki, H., Dhaliwa, N., Hussain, W., & Asif, M. (2019). Integrating design

thinking into extreme programming. Journal of Ambient. *Intelligence and Humanized Computing, 10, 2485-2492.*

Sommerville, I. (2011). *SOFTWARE ENGINEERING*.

Soomro, A. B., Salleh, N., Mendes, E., Grundy, J., Burch, G., & Nordin, A. (2016). *The effect of software engineers' personality traits on team climate and performance: A Systematic Literature Review. Information and software technology*. *73*, 52–65.

Srivastava, A., Bhardwaj, S., & Saraswat, S. (2017). SCRUM model for agile methodology. *In 2017 International Conference on Computing, Communication and Automation (ICCCA) (Pp. 864-869). IEEE.*

Stamm, S. (2023). Desired Qualifications Sought in Entry Level Software Engineers . In *Proceedings of the 54th ACM Technical Symposium on Computer Science* (pp. 854–860). Association for Computing Machinery . https://doi.org/10.1145/3545945.3569849

Standifer, R. L., & Lester, S. W. (2020). Actual Versus Perceived Generational Differences in the Preferred Working Context: An Empirical Study: Research. *Journal of Intergenerational Relationships*, *18*(1), 48–70. https://doi.org/10.1080/15350770.2019.1618778

Storey, M. A., Ernst, N. A., Williams, C., & Kalliamvakou, E. (2020). The who, what, how of software engineering research: a socio-technical framework. *Empirical Software Engineering*, *25*(5), 4097–4129. https://doi.org/10.1007/s10664-020-09858-z

Stypińska, J., Rosales, A., & Svensson, J. (2023). Silicon Valley ageism–ideologies and practices of expulsion in the technology industry. In *Routledge*. Routledge. https://doi.org/10.4324/9781003323686

Suleman, F. (2021). Revisiting the concept of employability through economic theories:

Contributions, limitations and policy implications. *Higher Education Quarterly*, *75*(4), 548–561. https://doi.org/10.1111/hequ.12320

Talarico, D. (2020). Market effectively to adult learners by understanding generational differences. *Recruiting & Retaining Adult Learners*, *22*(5), 1–3. https://doi.org/10.1002/nsr.30560

Tam, C., Moura, E. J. da C., Oliveira, T., & Varajão, J. (2020). The factors influencing the success of on-going agile software development projects. *International Journal of Project Management*, *38*(3), 165–176. https://doi.org/10.1016/j.ijproman.2020.02.001

Tam, H. lin, Chan, A. Y. fung, Fung, T. T. on, & Isangha, S. O. (2024). The mediating effect of psychological strengths and resilience on enhancing youth employability through social entrepreneurship education and training. *Children and Youth Services Review*, *156*(July 2021), 107325. https://doi.org/10.1016/j.childyouth.2023.107325

Tenzer, H., & Yang, P. (2020). The Impact of Organisational Support and Individual Achievement Orientation on Creative Deviance. *International Journal of Innovation Management*, *24*(2), 1–28. https://doi.org/10.1142/S1363919620500206

Tirado, J. M. (2021). *Why Do Developers Hate Programming Languages?* https://betterprogramming.pub/why-developers-hate-programming-languages-3955676083be

Trotman, K. T. (1996). Research Methods for Judgement and Decision Making Studies in Auditing. *Melbourne: Coopers and Lybrand*.

Turner, A. M., & Gurenlian, J. A. R. (2023). A comparison of Generation Z and Millennial dental hygiene students' preferred learning styles. *International Journal of Dental Hygiene*, *July*, 691–698. https://doi.org/10.1111/idh.12727

Urick, M. (2017). Adapting training to meet the preferred learning styles of different generations. *International Journal of Training and Development*, *21*(1), 53–59. https://doi.org/10.1111/ijtd.12093

Utsch, A., & Rauch, A. (2000). Innovativeness and initiative as mediators between achievement orientation and venture performance. *European Journal of Work and Organizational Psychology*, *9*(1), 45–62. https://doi.org/10.1080/135943200398058

Vadlamani, S. L., & Baysal, O. (2020). Studying Software Developer Expertise and Contributions in Stack Overflow and GitHub. *Proceedings - 2020 IEEE International Conference on Software Maintenance and Evolution, ICSME 2020*, 312–323. https://doi.org/10.1109/ICSME46990.2020.00038

Vaismoradi, M., Turunen, H., & Bondas, T. (2013). Content analysis and thematic analysis: Implications for conducting a qualitative descriptive study. *Nursing and Health Sciences*, *15*(3), 398–405. https://doi.org/10.1111/nhs.12048

Van Aken, J. E., & Berends, H. (2018). Problem solving in organizations. *Cambridge University Press*.

Veiga, A. P. (2017). Project success in Agile development projects. *ArXiv Preprint ArXiv:1711.06851*.

Venters, C. C., Capilla, R., Betz, S., Penzenstadler, B., Crick, T., Crouch, S., Nakagawa, E. Y., Becker, C., & Carrillo, C. (2018). Software sustainability: Research and practice from a software architecture viewpoint. *Journal of Systems and Software*, *138*, 174–188. https://doi.org/10.1016/j.jss.2017.12.026

Voorhees, R. A. (2001). Competency-Based Learning Models: A Necessary Future. *New Directions for Institutional Research*, *2001*(110), 5–13. https://doi.org/10.1002/ir.7

Wagenaar, R. (2014). Competences and learning outcomes: a panacea for understanding the (new) role of Higher Education? *Tuning J. High. Educ.*, *1*, 279–302. https://doi.org/10.18543/tjhe-1(2)-2014pp279-302

Wahyuni, D. (2012). The Research Design Maze: Understanding Paradigms, Cases, Methods and Methodologies. *Journal of Applied Management Accounting Research, 10(1), 69–80. Https://Doi.Org/10.1675/1524-4695(2008)31*.

Wang, C., Cui, P., Daneva, M., & Kassab, M. (2018). Understanding what industry wants from requirements engineers: An exploration of RE jobs in Canada. *International Symposium on Empirical Software Engineering and Measurement*. https://doi.org/10.1145/3239235.3268916

Waziri, M., Ngaruko, D., & Ngatuni, P. (2024). Influence of Entrepreneurship Knowledge on Employability of TVET Graduates in Tanzania: The Moderating Role of Self-Efficacy. *East African Journal of Business and Economics*, *7*(1), 407–425. https://doi.org/10.37284/eajbe.7.1.2134

Weimar, E., Nugroho, A., Visser, J., Plaat, A., Goudbeek, M., & Schouten, A. P. (2017). *The Influence of Teamwork Quality on Software Team Performance*. 1–33. http://arxiv.org/abs/1701.06146

Williams, C. (2007). Research methods. *Journal of Business & Economics Research (JBER), 5(3)*.

World Bank Group. (2019). *Ghana Digital Economy Diagnostic*. https://doi.org/10.1596/34366

Wu, G., Liu, C., Zhao, X., & Zuo, J. (2017). Investigating the relationship between communication-conflict interaction and project success among construction project

teams. *International Journal of Project Management*, *35*(8), 1466–1482. https://doi.org/10.1016/j.ijproman.2017.08.006

Xie, B., Loksa, D., Nelson, G. L., Davidson, M. J., Dong, D., Kwik, H., Tan, A. H., Hwa, L., Li, M., & Ko, A. J. (2019). A theory of instruction for introductory programming skills. *Computer Science Education*, *29*(2–3), 205–253. https://doi.org/10.1080/08993408.2019.1565235

Yawson, D. E., & Yamoah, F. A. (2020). Understanding pedagogical essentials of employability embedded curricula for business school undergraduates: a multi-generational cohort perspective. *Higher Education Pedagogies*, *5*(1), 360–380. https://doi.org/10.1080/23752696.2020.1846134

Yazdanian, R., West, R., & Dillenbourg, P. (2021). Keeping up with the trends: Analyzing the dynamics of online learning and hiring platforms in the software programming domain. *International Journal of Artificial Intelligence in Education*, *31*, 896-939.

Ye, X., Shen, H., Ma, X., Bunescu, R., & Liu, C. (2016). From word embeddings to document similarities for improved information retrieval in software engineering. *Proceedings - International Conference on Software Engineering*, *14-22-May-*, 404–415. https://doi.org/10.1145/2884781.2884862

Yin, R. K. (1998). *The abridged version of case study research*. Handbook of applied social research methods.

Yin, R. K. (2003). *Designing case studies*. Qualitative Research Methods.

Yin, R. K. (2012). *Case study methods*.

Yin, R. K. (2018). Case Study Research and Applications. In *Thousand Oaks, CA: Sage* (Vol. 6). https://doi.org/10.7222/marketing.2023.045

Yu, J. (2018). Research Process on Software Development Model. *IOP Conference Series: Materials Science and Engineering*, *394*(3), 0–8. https://doi.org/10.1088/1757-899X/394/3/032045

Zhang, H., Babar, M. A., & Tell, P. (2011). Identifying relevant studies in software engineering. *Information and Software Technology*, *53*(6), 625–637. https://doi.org/10.1016/j.infsof.2010.12.010

Zhang, N., & Gu, L. (2023). *Demand analysis and training scheme research of big data talents based on Python*. *12510*(Csdsci 2022), 3. https://doi.org/10.1117/12.2656685

Zhi, Q., & Morisaki, S. (2021). An evaluation of value-oriented review for software requirements specification. *Computer Systems Science and Engineering*, *37*(3), 443–461. https://doi.org/10.32604/CSSE.2021.015157

APPENDICES

Appendix A – Research Interview Guide

INTERVIEW GUIDE – FOR SOFTWARE ENGINEERS

Dear Participant:

I am David Boadi, an MPHIL student of the University of Ghana Business School pursuing Management Information Systems. I am conducting a study on the topic: "The influence of Software Engineering Competencies on Employment: Does knowledge in Programming Languages and Generational Age matters?"

The specific objectives of the study are:

1. To understand software engineering competencies required by employers in Ghana.
2. To explore the impact of software engineering competencies on employment in Ghana.
3. To investigate the moderating influence of generational age and knowledge in programming languages on software engineering competencies and employment.

I would greatly appreciate it if you could spare a few minutes from your busy schedule to complete this interview. Your candid responses will be highly appreciated. Please be rest assured that the information to be gathered from you is purely intended for academic purposes only. Your involvement in this study is completely voluntarily, and you retain the right to abstain from responding to any question or to withdraw from the research at any moment.

For any enquiry, you may reach me via email at dboadi013@stu.ug.edu.gh or my supervisor at eakolog@ug.edu.gh. Thank you.

Date: ……

SECTION A – Background of Respondent

1. Please tell me about yourself, including your current position and unit within your organization as well as your highest educational qualification.
2. How many years of experience do you have in software engineering?
3. Kindly describe the type of organization you are currently working with as a software engineer. For example, is it a software development company, an educational institution, a financial institution, or any other?
4. What specific software engineering domain does your organization specialize in, such as Website Development, Mobile Application Development, Desktop Application Development, Blockchain, Machine Learning, or any others?
5. Which specific software engineering tasks do you perform in your organization?
6. How long have you been leading/managing software engineers?
7. Could you please describe some of the software engineering positions that exist within your organization?

SECTION B

Competency Identification

8. What technical competencies are expected working as a software engineer in your organization?
9. What soft competencies are expected working as a software engineer in your organization?
10. Kindly give a brief description of the competencies.
11. Why are the competencies you have mentioned important?

Influence of Software Engineering Competencies on Employment

12. What is the importance of the following competencies for software engineering roles within your organization?

    i) Achievement Orientation

    ii) Domain Knowledge

    iii) Communication Skills

    iv) Problem Solving Ability

    v) Professional Work Ethic Ability

    vi) Teamwork Ability

13. Kindly describe how the various competencies mentioned are assessed during your recruitment and selection of software engineers.
14. Can you explain how the various competencies mentioned affect employment opportunities for software engineers within your organization?

The Moderating Influence of Knowledge in Programming Languages and Employment

15. How important is knowledge in programming languages for software engineering roles at your company?
16. Can you describe how proficiency in programming languages is evaluated during the hiring process for software engineers?
17. In what ways does knowledge in programming languages enhance the domain knowledge of software engineers in your organization and how does this interaction influence their employment opportunities and career advancement?

18. In what ways does knowledge in programming languages enhance the problem-solving abilities of software engineers in your organization, and how does this interaction impact their employment opportunities and career advancement?

The Moderating Influence of Generational Age and Employment

The questions in this section pertain to generational age (Gen X, Y, Z). In the context of this study, Gen X are persons born between 1965 and 1981 (43 - 59 years old). Gen Y were born between 1982 and 1999 (25 - 42 years old) whereas Gen Z were born between 2000 and 2012 (12 - 24 years old).

19. Does a generational mix of software engineers (Gen X, Y, Z) offer any advantages to software firms? Kindly explain your answer.
20. Have you noticed differences in domain knowledge across the various generational cohorts (Gen X, Y, and Z)? If so, kindly explain how they influence the employment outcomes of software engineers.
21. In your experience, do the various generational cohorts (Gen X, Y, and Z) exhibit different communication skills? If so, how do they influence the employment outcomes of software engineers?
22. Have you observed differences in problem-solving abilities across the various generational cohorts (Gen X, Y, and Z)? If so, how do they influence the employment outcomes of software engineers?

Closure

23. I am done with my questions. Is there anything you would want to add or clarify?
24. Are there any documents that you would like to share with me which may contribute to this study?

Please accept my sincere gratitude for your time and thoughtfulness. *Before the write-up is finalized, I will transcribe your responses and send them to you for clarification if needed.*