A Software Process Engineering Approach to Understanding Software Productivity and Team Personality Characteristics: An Empirical Investigation

Murat YILMAZ

Master of Science in Software Engineering

A Dissertation submitted in fulfilment of the
requirements for the award of
Doctor of Philosophy (Ph.D.)

to the

Dublin City University

Faculty of Engineering and Computing, School of Computing

Supervisor: Dr. Rory V. O'Connor

January 2013

Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Doctor of Philosophy is entirely my own work, that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge breach any law of copyright, and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed: _____

ID No.: 59111500

Acknowledgements

Although intellectually a fruitful experience, working on a PhD may be one of the most challenging undertakings I have ever had in my whole life. This is mostly because, PhD research requires a sustained and intense focus on your project; learning to write academic papers, reading them over and over again, presenting your work, and staying up till the early-morning light. Important accomplishments are rarely product of a lonely process; a researcher needs a group of people who believe in his success.

During the course of this unforgettable experience, I am indebted to a number of people who did not leave me alone while I was lost amid confusion. First and the foremost, I am deeply grateful to my supervisor Dr. Rory V. O'Connor, who mentored me through every step of my doctoral studies with his remarkable communicative and academic skills. His ability to set high quality standards, his vision, patience, wit and wisdom that support and guide me through every stumble in the steps of my PhD project made this thesis possible.

My motivation to pursue a PhD degree in Software Engineering was initiated while I was working on my Masters degree at the University of Minnesota. Henceforth, I would like to thank Dr. John Collins, my Masters supervisor, who helped me to shape my preliminary idea. In addition, I would like to extend a heartfelt thanks to Dr. Ugur Halici, and Bulent Kandiller, to whom I cut my teeth in several different projects in the software industry.

I am also grateful to Lero - the Irish Software Engineering Research Center for funding my doctorate and to Dublin City University for the numerous facilities they provided. It has been a privilege to get to know both the people at DCU and my colleagues at Lero.

Unfortunately, over the course of my life, I have lost a number of people close to heart. Among them, I am quite certain that my mother Yucel and my grandmother Rana would be proud of me for pursuing a doctoral degree. May they all rest in peace!

Many thanks to my colleagues and close friends; Ozgur Ergul, Murat Yeralti, Berke Atasoy, Pelin Atasoy, Burak Elmas, Demokan Atasoy for exchanging ideas, sharing visions, providing me with their constructive comments and vivid suggestions for several parts of this study. Furthermore, I would like to extend my sincerest thanks to Paul Clarke for his efforts to improve the quality of my Ireland experience particularly with his supportive companionship in the form of a whimsical elf.

Thanks to Bulent Ozen, Mr. Atalay, Y. M. Erten, Nagihan, Emrah, Alper and Musa for their support in the industrial settings. It was greatly appreciated.

Finally, I save the last and the most insufficient word of gratitude for my wife, *Aysel*, not only for her love but also her valuable insights, and her ultimate support and encouragement. Concurrently, I deeply appreciate her unwavering patience for the times I was away from home to pursue my PhD. Thank you with all my heart!

I would like to dedicate this work to Aysel Yilmaz for her loving and support, which makes the PhD her success as much as it is mine.

Science is the most reliable guide for civilization,
for life, for success in the world.
Searching a guide other than the science is
meaning carelessness, ignorance and heresy.

Mustafa Kemal Atatürk

Contents

List of Figures

List of Tables

Abstract

Despite the significant effort in managing the complexities of software development by using several engineering analogies, there is still no comprehensive approach that recognizes software development as a social activity and software productivity in form of an intangible asset gained and maintained by social relations. This study proposes a model in which software productivity improvement is investigated as a function of the factors affecting productivity, whereas team productivity is considered as a compatibility problem of distinct personality traits that should be situated in an efficient social configuration. The fundamental assumption is that the productivity is a latent construct measurable by a set of indicators and improvable by relating personality traits and team configurations.

The two main contribution of this thesis is to develop an understanding of the factors affecting software productivity by empirical investigation and to build a personality-profiling test based on a psychometric scale specific to software practitioners. To assess the validity of our approach, we conducted a two-step empirical study in an industrial setting: (i) a psychometric survey on 216 software practitioners for measuring the correlations among the factors affecting productivity, and (ii) a domain specific game-based personality test on 63 participants for investigating the implications of personality types over the effective team configurations. Our findings indicate that software productivity is positively associated with social productivity and social capital, and can be measured by 21 different indicators identified from the literature. In addition, social aspects such as team size and individual's characteristics have a significant affect on productive team formations. A strong negative association is observed between social capital and the time practitioners spend in a software company. Evidence suggests that individuals in software teams become more extroverted while the effective configurations are still achieved with teams populated by balanced personality traits.

Related Peer Reviewed Publications

Twelve papers (numbered P1 to P12) and two posters directly are related to the PhD research have been published are as follows:

Papers

Yilmaz M., O'Connor R.V., "Maximizing the Value of the Software Development Process by Game Theoretic Analysis: A proposal for research into Game Theory" in Proceedings of the 11th International Conference on Product Focused Software, ser. PROFES 10. New York, NY, USA: ACM, 2010, pp. 9396. **(P1)**

Yilmaz M., O'Connor R.V., Collins J., "Improving Software Development Process through Economic Mechanism Design", EuroSPI 2010 (European Software Process Improvement Conference), A. Riel et al, Eds., vol. 99. Springer Berlin Heidelberg, 2010, pp. 177-188. **(P2)**

Yilmaz, M. and O'Connor, R., "An Empirical Investigation into Social Productivity of a Software Process: An approach by using the structural equation modeling", 18th European Software Process Improvement Conference, CCIS Vol. 172, Springer-Verlag, June 2011. **(P3)**

Yilmaz, M. and O'Connor, R., "An approach for improving the social aspects of the software development process by using a game theoretic perspective: towards a theory of social productivity of software development teams", 6th International Conference on Software and Data Technologies, July 2011. **(P4)**

Yilmaz, M. and O'Connor, R., "A Software Process Engineering Approach to Improving Software Team Productivity using Socioeconomic Mechanism Design ", ACM SIGSOFT Software Engineering Notes, Vol. 36, No. 4, September, pp. 1-5, 2011. **(P5)**

Yilmaz, M. and O'Connor, R., "Using Game Theory to Improve Productivity Software Teams - Developed a Software Process Engineering Approach [in Turkish]", Proceedings Turkish National Software Engineering Symposium, September, 2011. **(P6)**

Yilmaz, M. and O'Connor, R., "A Systematic Approach to the Comparison of Roles in the Software Development Processes, Proceedings 12th International Conference on Software Process Improvement and Capability dEtermination", CCIS Vol. 290, Springer-Verlag, May 2012. **(P7)**

Yilmaz, M. and O'Connor, R., "A Market Based Approach for Resolving Resource Constrained Task Allocation Problems in a Software Development Process", 19th European Conference on Systems, Software and Services Process Improvement (EuroSPI 2012), CCIS Vol. 301, Springer-Verlag, June 2012. **(P8)**

Yilmaz, M. and O'Connor, R., " Towards the Understanding and Classification of the Personality Traits of Software Development Practitioners: Situational Context Cards Approach", 38th Euromicro Conference on Software Engineering and Advanced Applications, September 2012. **(P9)**

Yilmaz, M. and O'Connor, R., "Social Capital as a Determinant Factor of Software Development Productivity: An Empirical Study using Structural Equation Modeling", International Journal of Human Capital and Information Technology Professionals, Vol. 3, No. 2, 2012. pp. 40-62. **(P10)**

Related Publications: Under Peer Review

Yilmaz, M. and O'Connor, R., "An Empirical Study to Evaluate the Factors Affecting Software Development Productivity", Under review for the Journal of Empirical Software Engineering. Submitted December 2012. **(P11)**

Yilmaz, M. and O'Connor, R., "Towards A Game-Based Methodology for Understanding Effective Software Team Structures". Under review for Journal of Systems and Software. Submitted October 2012. **(P12)**

Posters

Yilmaz, M. and O'Connor, R., "Improving software process productivity by modeling the interaction among the team social structures using game theoretic analysis", Poster, SEPG Europe Conference, Dublin, Ireland, June 2011.

Yilmaz, M. and O'Connor, R., "Improving Software Development Process through Economic Mechanism Design", Research Poster at Lero Research Showcase, 2010.

Part I

Preliminaries

Chapter 1

Introduction

Software process and productivity improvement encompass the activities, which promise to increase the quality of a software product [1]. Predictably, these activities need to align process, tools and technologies with human factors and social considerations [2, 3]. According to DeMarco and Lister [4], the major problems encountered in software development activities are more sociological than technical in their nature. Software development is a form of social activity [3, 5]. Therefore, it is commonly conducted by teams consisting of individuals identified by characteristics of individualism, rationality, and mutual interdependence [6]. In this particular viewpoint, one can argue that several factors affecting the software development process should arise from the complexity of individuals' interactions and social communication costs. Because they are hindering the software productivity, the investigation of social factors and corresponding interest in social aspects of software development has become a part of software engineering body of research [5].

Software development is mainly governed by human-centric activities, and therefore, a number of social factors and individuals' personalities should have a significant impact on the productivity of software development as well as on the effectiveness of software development teams. Although it is becoming increasingly difficult to ignore the importance of social aspects of software development for productivity improvements, to date most of the earlier software productiv-

ity research has focused heavily on the software process with little importance attached to the impact of the personalities of the individuals involved.
Recently, the social issues of software development have received critical attention [7]. Thus, there is a vital need to understand both the factors of software development productivity and effective software team configurations, which are at the heart of understanding the human aspects of the software development process. However, there is still a gap between the social world of software development and the technical world of software development landscapes [5].
To address this gap, this research has adopted a novel multi-perspective approach to examining the factors and the personality characteristics of software practitioners affecting the productivity of software development organizations. Firstly, the factors affecting the software productivity are investigated and several techniques are applied to identification of these factors. Secondly, software practitioners' personalities and their preferences are investigated in detail by using a game-based approach. This is the first published research to demonstrate such findings, which are emphasizing the importance of human considerations in software development.

1.1 Research Problem

Despite the significant varieties of software development processes and project management techniques, software projects may still fail for a variety of reasons: (i) vague definitions of project goals, (ii) imprecision of the resource estimates, (iii) communication and coordination problems (e.g. conflicts) among the stakeholders, and (iv) bad managerial decision making [8]. Hartman [9] reports that the percentage of failed projects has significantly declined, however, many software projects are still unsuccessful for several reasons such as limited communication among the individuals and a lack of appropriate method for measuring software productivity. Jones states in his book "*As of 2009, the great majority of companies and the great majority of software engineers have no effective measurements of either productivity or quality.*" [10, pp. 20]. The available evidence confirms that a number of software project failures are not

only because of technical difficulties but also because of human related factors such as team incompatibility problems, and personal conflicts [4].

As regards people, their interactions, and the impact of software organization on productivity, it is important to understand that a substantial amount of these problems originate from social issues [3, 11]. Consequently, software development productivity seems to be affected by (i) factors affecting software productivity and its social aspects, (ii) personality characteristics of participants that are involved in the development process.

In addition, we envision that the outcomes of a software production process should heavily rely on the classification of the psychological and social structure of individuals regarding their roles and characteristics among the development organization. For the individuals working in such networks, we should consider methods that support the maximization of their collective output. Such methods should generally involve profiling the personality characteristics of the individuals that constitute the network and illustrating their efficient team structures based on their personality types, and ultimately seek ways to maximize the productivity of the group as a whole. However, such techniques are currently underutilized in software development.

This project was conceived while I was working in the software development industry. As a senior software practitioner, I often observed software productivity problems, which were due to not only technical factors but also the social issues such as team staffing, team size, and member compatibility. The fundamental motivation of this study is a belief that it is important to reveal the software productivity factors of a software development organization and explore effective software team configurations by visualizing their personality characteristics as a whole. Industrial evidence found concerning the subject matter is presented in this thesis in Part IV Empirical Contributions.

1.2 Context of Current Research

This section details the characteristics of the research domain. Firstly, the importance of the value-based understanding of the factors affecting software development productivity is discussed. Secondly, the benefits of a game based personality assessment specifically developed for software development landscapes are presented.

1.2.1 Value-Based Software Development Productivity

Software engineering is generally considered as a challenging team based activity, which requires the interaction of one or more skillful individuals. Such a viewpoint suggests that it should be performed in the form of a social activity to promote its collaborative and social aspects within a wide spectrum of stakeholders [12, 13]. Indeed, over the last decade a significant amount of software engineering researchers have considered the software practice as a *social activity* and have conducted research on the implications of social approaches to the software development process [5]. Being able to create a quantifiable value (e.g. valuation of production assets, managing decisions under uncertainty, etc.) from the planned software activities not only accomplishes stakeholders' goals but also is an axiomatic step towards improving the productivity of software development organizations as a whole. However, it is the proposition of the research that understanding the social problems of software development requires a different management approach, which can characterize the flow of knowledge, its role in decision-making, and their consequences, which relate the individual and collective goals of the stakeholders [14]. As a consequence of indefinite scientific definition of quality in software engineering literature, it appears that there is a lack of understanding of the economic value of software quality, which seems to be an important referent that is hindering software productivity [10]. In fact, the present study contends that there is a need for techniques to deal with the factors that hinder the velocity (i.e. productivity) of software development.

Despite the fact that it might be easier to organize the software development

work in a value neutral environments, this has been found as one of the potential reasons for the *failure of software projects* in several development landscapes [15]. To bridge this gap, a number of approaches to integrating value-based propositions in software engineering research have been adopted. Dating back to 1980s, Boehm introduced the first idea of integrating value considerations for the activities of software development in his book, *Software Engineering Economics* [16]. Later, research was performed to apply the concepts from economics to software engineering. For example, some studies focused on addressing problems in software architecture [17] and design decisions as real options [18], understanding software development as an investment activity [19], analyzing risks and uncertainties that alters the value of information [20], improving the speed and therefore the productivity of the software development process [21], and constructing several value-based software engineering frameworks [14, 22, 23], etc.

One of the main considerations of value-based software development productivity is to understand the people management issues of software development using a value-oriented approach, useful for managing stakeholder negotiations, building software teams and maintaining team configurations [22]. It can be used for investigating social and ethical issues that are encountered in the software development life-cycle. These managerial issues, however, could cause fluctuations in the productivity of a software development process. Moreover, it is important to recognize that the effectiveness of software development teams is also dependent on the factors that are affecting the software development productivity.

To understand the implications of the productivity framework of a software organization, a series of models have been proposed with emphasis on social and economic factors inducing productivity. Thus, we claim that empirically validating the factors based models that are contributing to the software development productivity will provide an efficient way to maximize the economic value created in a software development process.

1.2.2 Games Playable by Software Teams

In the last decade, we have learned that the technical skills of people are not enough to construct highly productive software teams (see e.g. [24, 25]). One reason for this is that the *new* knowledge-driven global economy shifts its demands from technical to social skills of individuals [26], who can adopt networked forms of an organization [27]. Recently, a substantial amount of literature has reported that software development is a social activity [3, 5], in which individuals with diverse personal characteristics are to work together to produce sophisticated *software artifacts* [28]. Ryan and O'Connor [29] suggest that software development relies on the communication skills and abilities of a software development team, and therefore the productivity of a software organization is positively affected by the levels of social cohesion among its members. Games can be used to improve the social cohesion, and they reveal individuals' behaviors.

Game theory is a field of mathematics, which has been frequently applied to social sciences (especially in sociology and political sciences) for analyzing many different situations, e.g. variability of individual behaviors, and formation of coalition structures among the individuals and human networks [30]. Furthermore, the study of games has flourished over the last several decades and attracted many researchers. As a result, it has been applied to several diverse fields [31] including biology, linguistics, psychology, philosophy, and later in computer science [32]. The evidence exists, suggesting that software development organizations rely heavily on the *strategic nature* of software management activities [33,34] (e.g. identifying stakeholders, understanding competitors and market conditions). Therefore, we suggest that software management teams should benefit from what game theory offers.

The notion of games is ubiquitous and highly connected with characterization of human activities with several benefits. They could even offer solutions for society based problems in a social setting [35]. Therefore, it is not surprising to discover that games that are played in societies have great social and economic benefits. From the outset, games enable us to form collective social

organizations, which ultimately produce considerable advantage for building complex software artifacts. On the other hand, to constitute better software development teams, we should benefit from personality assessments, which may also particularly be useful for improving the team productivity in the software development process.

A general assumption is that personality traits significantly affect the human behavior [36]. Although most people agree types of behaviors vary in different situations, for most of the individuals there is an observable pattern of consistency in their behaviors. The classical approach to organizing individuals according to their personality types requires performing a personality questionnaire. Several psychometric tests are used to capture non-context dependent behavior. For example, they are used to assign personnel to the right job, by predicting potential skills. Same of these tests are primary mental abilities test, wonderlic personnel test, and programmer aptitude test [37]. However, observing individual's verbal decisions in several different context-dependent situations can more effectively reveal the personality traits of an individual.

The personality profiling of software practitioners is a challenging task. There are contradicting arguments (e.g. [38]) about the difficulty of revealing the personality traits of a software practitioner or an IT project personnel using a classical a psychological-assessment test such as the Myers-Briggs Type Indicator (MBTI). Some personality researchers also suggest that a verbal report is not enough to evaluate their success, while others claim that these tests are powerful enough to reveal participants' personality types. In practice, however, there is no assurance that the answers of individuals are accurate.

Being highly interactive, a game-based alternative provides a real advantage over classical approaches. Not only does it strengthen the social fabric of a software development team, but it also improves collaboration skills and individuals' concentration on collective outcomes. Additionally, games have a potential to keep people individually motivated [35] with the idea of social success and improve their ability to understand each others' thoughts and feelings. The idea of creating a card game for personality profiling stems from the fact

that traditional personality tests may have some inadequacies [39]. To improve the accuracy of the results, we suggest that the type of an individual can be retrieved by using events or situations derived from the software domain instead of using the conventional MBTI questionnaire. Kaluzniacky's words eloquently support this: "*Eventually, perhaps an actual IT personality diagnostic instrument could be developed containing only questions with an IT work context, but paralleling closely the questions in the MBTI itself.*" [40, pp. 55].

1.3 Research Objectives

The first part of this research proposes an empirical approach to investigating the relationships among the hypothetical latent constructs[1] based on the factors[2] that are affecting software development productivity - a technique that can be used to evaluate the conceptual propositions with respect to the accuracy of data collected. First, we hypothesize the relationships between several social and economic determinants identified in the literature potentially affecting the productivity of software development. Based on the identified factors, we build several advanced models, and test them with data collected from a software development organization. Next, we analyze these models by using a series of statistical techniques such as factor and path analysis. Moreover, based on the data collected, in the second part of this assessment, we analyze the impact of software roles and team constructs on the latent factors affecting software development productivity. Thirdly, we develop a game-based approach to portray the personality traits of a team of practitioners on a psychometric scale, and quantify the personality traits to understand the compatible traits and to maximize the productivity of software development teams by building better team configurations.

[1] A conceptual variable that can not be either observed or measured directly.
[2] To measure and present a latent construct a set of observable indicators are captured.

The objectives of the research are as follows:

> **Objective 1:** Measure the relationships among several productivity factors and their associations with the latent constructs (i.e. productivity, social productivity and social capital) as identified in the literature through a confirmatory factor analysis model.
>
> **Objective 2:** Explore the impact of teams and software development roles on productivity, social productivity, and social capital.
>
> **Objective 3:** Build a game-based MBTI-like survey instrument specific to the software engineering domain, which can reveal and illustrate software practitioners' personality characteristics.

1.4 Research Questions and Hypotheses

In response to the issues highlighted above, in this section, we develop a list of research questions, which guide the research. All of the proposed hypotheses are used to evaluate our conceptual propositions with respect to the empirical data collected. We seek answers to our research questions by analyzing the data by using systematic and rigorous approaches that are specifically tailored for this study.

> **Research Question 1:** *Can we quantify productivity by using a set of indicators and with the latent constructs (e.g. social capital and social productivity) that are potentially affecting productivity?*
>
> **Research Question 2:** *Can a positive correlation between productivity, social productivity and social capital be measured for software development?*

To date, as it is qualitative in its nature, software productivity as a notion has been found hard to measure [10]. In the light of this argument, we hypothesize a model of productivity in terms of social productivity and social capital.

Ultimately, the goal of the first research question is to identify the relationship between latent variables that we construct and the observable variables for each latent construct that was found in the literature. The second research question seeks a correlation between two latent constructs: productivity and social productivity based on the identified indicators. This part of the research is concerned with identifying the relationship between productivity and its potential aspects, and hence the first hypothesis has been developed to support this endeavor.
To seek answers to these questions, we have established and formalized our first hypothesis guided by our research agenda.

> **Hypothesis 1:** *There is a significant correlation between the factors affecting software development and the productivity of software development.*

The second set of research questions intend to uncover the ways to improve team productivity using appropriate parameters such as the actual and ideal size of a team for better productivity, the years a practitioner spend in a company, years of experience. Furthermore, we seek out a significant relationship between these variables, our latent constructs and the roles that practitioners are assigned during the course of the development activities.

> **Research Question 3:** *Can we observe a relationship between roles of software practitioners and the observed team productivity?*

> **Research Question 4:** *Is there any empirical relationship between social capital and identified variables to measure the variations in software team productivity?*

The second hypothesis relies on the argument that our hypothetical constructs such as social capital are related with the identified variables that are potentially affecting the productivity of software development. To seek answers to these questions, we have established our second hypothesis:

Hypothesis 2: *There is an observable relationship among the perceived team productivity, roles and our hypothetical (latent) constructs of software productivity*

The next group of research questions aim to facilitate a game-based approach to personality traits identification of software practitioners. The expected achievement here is to find techniques to identify the personality traits with a game-based approach. Consequently, the goal is to empirically understand the software team structures in terms of their personality traits so as to gain an ability to predict effective and productive team formations. To this end, we should be able to visualize the software teams with respect to practitioners' personality types and explore their structures. For example, by investigating and visualizing the social structure of a team, (i) personality characteristics of the participants can be identified, (ii) the overall personality of a team could be analyzed, (iii) the outcome of the interaction of different practitioners can be outlined.
In order to understand the research problem, we suggest that two questions need to be asked:

Research Question 5: *Can we reveal the personality traits of software practitioners by using a context specific, game-like profiling method?*

This question is about profiling the personality characteristics of software practitioners by using a novel technique, i.e. game playing.

Research Question 6: *Can we build a visualization instrument to illustrate software team personality types?*

At this point, the following question should be asked: Does revealing the personality characteristics of the participants have a positive impact on building effective team configuration? Therefore, the previously known productive team formations should be investigated for specific patterns of personality type combination or constraints. To examine the overall personality trait of a team,

the present study suggests illustrating the traits of individuals on a software development team on a graph using a novel form of visualization.

> **Hypothesis 3:** *Personality characteristics of individuals in software development teams can be revealed and illustrated by using a context specific game-based profiling technique.*

Based on a knowledge-based production economy, we consider that the productivity of software teams is significantly affected by personality traits of its members. Consequently, the third hypothesis states that an empirical analysis is necessary to reveal the personality types of individuals and single out the best possible combinations to understand effective team configurations. To this end, the team design space should be expanded from technical aspects to include the social influence. The goal here is to travel beyond the capabilities of current team building methods especially designed for software teams.
The third hypothesis suggests that adopting a human-centric view as a complement to existing process focused approaches has benefits. We envision that if we illustrate the personality traits of a software development team as a whole, to a certain extent, we can identify *team personality* characteristics, which could be useful for understanding effective team structures. Secondly, among others, we shall seek answers to questions like *"Are the individuals gravitate to specific roles based on their personality traits?"*

1.5 Organization of the Thesis

The overall structure of the study takes the form of five parts and ten chapters. The first part consists of two chapters. *Chapter 1* is the introductory chapter. It presents the motivational problem, thesis objectives, and research hypothesis. *Chapter 2* gives a preliminary background about the definition of the process, models and software process improvement.
The second part includes *Chapter 3*, which briefly reviews the research methods. It outlines the mixed-method research methodology, and justifies the research design, which will be followed by the sections on the existing literature of case

study, structural equation modeling, focus group study, grounded theory, and game playing as a research approach. Lastly, it details the research processes for two industrial case studies.

The third part begins by laying out the theoretical dimensions of the research in the form of theoretical contributions, which are presented in three chapters: *Chapter 4* reviews the theory of games in software engineering literature and mechanism design. It continues through an abstract game model. *Chapter 5* describes the social and value dynamics of software engineering that includes a review of software ecosystems, software artifact, productivity, value, social productivity, and social capital. *Chapter 6* surveys the roles, personality traits, and their applications in software engineering research.

The fourth part is related to practical (empirical) contributions, and the data collection processes undertaken during the course of this research. It presents the empirical findings of the two industrial case studies and their data analyses. *Chapter 7* starts with the first case study for identification of the factors affecting software development productivity, and the impact of teams and roles on the social constructs. *Chapter 8* represents the second case study about a game for revealing the personality types of software practitioners in order to illustrate software team structures.

The fifth part includes two chapters: *Chapter 9* provides discussions and limitations including a multi-view discussion about the results drawn upon the entire thesis, tying up the various theoretical and empirical strands, and finally *Chapter 10* summarizes the implications and contributions, and discusses the possibilities for future research.

Chapter 2

Background

2.1 Introduction

This chapter starts with a brief introduction to the notion of software development process. It continues with surveying software process models particularly focusing on spiral and WINWIN spiral models. Next, it details the concept of software process improvement, and reviews ISO/IEC 12207 and agile methods. The chapter concludes with a chapter summary.

2.2 Software Development Process

In the field of software engineering, various definitions of software development process can be found. Although differences of opinion still exist, there appears to be some agreement that a software process refers to a set or order of organizational activities (sometimes as a workflow) constrained with entrance and the exit criteria by man, machines and methods [41–43]. In addition, Feiler and Humphrey [44] consider the software development process as a set or sequence of steps followed to reach a defined goal, whereas Erdogmus [45] claims that a software process should be a set of patterns or activities compiled to find solution to a series of software development problems. ISO/IEC 12207 [46] defines the software development process as a "*set of interrelated or interacting activities, which transforms inputs into outputs*", where the standard embodies

the process as an outcome based instrument that should be beneficial to the stakeholders, Therefore, it should either finalize with an artifact production, or a change in state or, a solution to one of the designated limitations such as objectives, requirements.

A software development process explains the methods and procedures, in which organizations and individuals have to follow to create software products and services [47]. Ultimately, the goal of a software process is to provide a roadmap for the production of high quality software products that meets the needs of its stakeholders within a balanced schedule and budget [48]. It concentrates on creation and maintenance of tasks and activity structures rather than the output or the end product. Therefore, a typical software process should aim to solve the potential and future problems of software development with respect to planning and budgeting. In the context of this thesis,

> ***A software development process is considered as the coordination of structural social activities (e.g. management, production and maintenance) coupled and constrained with a set of individuals' (i.e. participants who perform the activities) roles and skills for producing software artifacts in a predefined productivity level.***

Despite the fact that sometimes it is not explicitly defined, all software development organizations use a form of process based on their beliefs, values, goals, or organizational skills [49]. A broad definition of software development process should encompasses development, deployment and maintenance of a software product, which should include organizational structures and policies (e.g. task definitions), human activities, technologies, and product functionalities [50, 51]. A meta-model for the abstraction of a software engineering process, its components, and a life-cycle model adopted from Unhelkar [52] is illustrated in Figure 2.1.

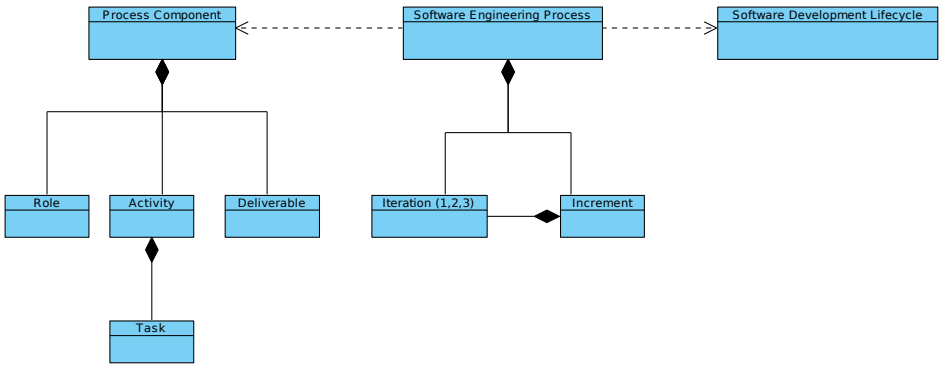


Figure 2.1: *A Meta-Model for Software Engineering Process.*

2.3 Software Process Models

Typically, a process model is used to refer to the activities of development and the ways to control the software product by understanding the steps taken throughout a process [17]. It represents an abstract representation of a process including the definition of a set of states or a sequence of activities that are created by rendering descriptions of the tasks of development, conceive their relations, and define the resulting outcomes [44].

Over the past few decades, a software process model and a life cycle model have been confused. However, Acuna *et al.* [3] resolve this confusion: The life cycle usually represents the steps that a software product should evolve through, which should be *specific* and *organization dependent*, whereas a process model is centered on the tasks or the activities performed for managing, developing and maintaining software systems rather than dealing with the inputs and the outputs of the production, and hence it should be *general* and *project dependent*. Many different variants of development models and methodologies have been proposed. Conventional depiction of a software process model includes the waterfall model [53], the iterative enhancement model [54], prototyping development model [55], the spiral model [56], WINWIN spiral model [57], and the agile methodologies including but not limited to extreme programming [58], scrum [59], and feature driven development [60]. In addition, the International Organization for Standardization (ISO) has developed the ISO/IEC 12207 [46]

standard for software lifecycle processes, which aims to be the standard that defines all the tasks required for developing and maintaining software, but which does not imply a specific lifecycle model. Instead, processes that are defined by this standard need to be outlined into a development model, and should be compromised within the organization [61].

2.3.1 Spiral Model

Boehm [56] proposes the idea of a risk-driven development and management model - *the spiral model*, which continually iterates over a series of development methods instead of a code-oriented or a document-driven approach. As a development process model, one of the main distinguishing characteristics among the other models is its aim to focus on development strategies dealing with project risk factors (e.g. prototyping is used as a method for mitigating the software development risks) [17]. In particular, it highlights the risks encircling software development and investment decisions, which are vitally important for development of large scale software and systems [62].

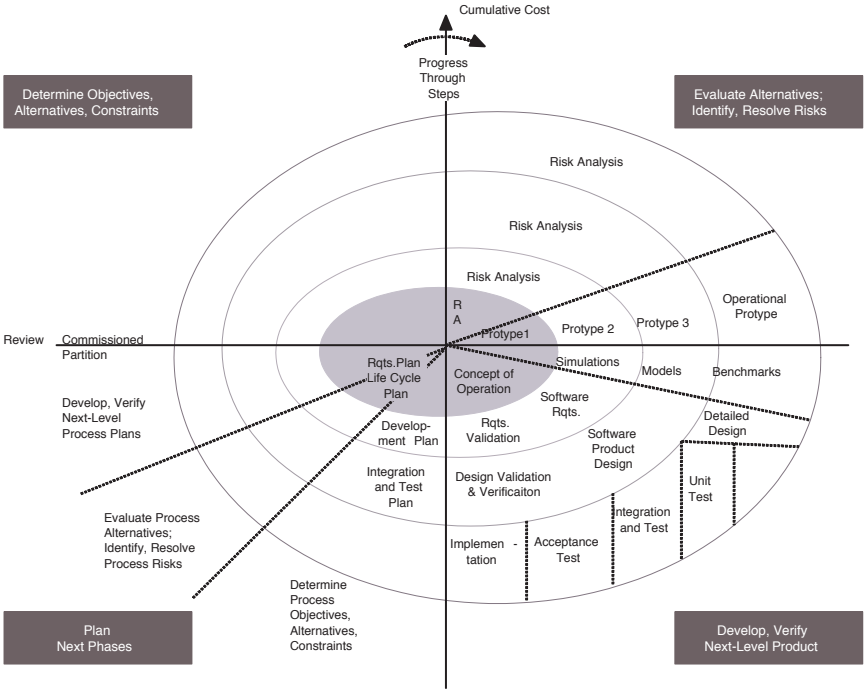


Figure 2.2: *The spiral development model*

Boehm created the original spiral model to include the aspects of earlier process models (e.g. waterfall, iterative, prototyping), creating a whirling spiral that supports the changing nature of products over time as depicted in Figure 2.2. The loops of the spiral represent different phases (Boehm's task regions); (i) improving developer and customer communication, (ii) planning for resource allocation, (iii) identifying the risks, and (iv) prototyping the application (v) building, testing and installing releases, (vi) evaluation of the software product [62]. Each spiral starts with identification of the goals for a part of the software, and continues with investigation of alternative methods (e.g. buying an off-the-shelf product or a module rather than developing), and finally, if necessary limitations of the alternatives are investigated to decide on a newer strategy [43].

The vision of dealing with volatility in software requirements by identifying the risk factors in early stages of development favors the spiral development as an important methodology especially for developing large software systems. Most importantly, it allows the software practitioners to develop a prototype at any stage regarding their needs [62]. However, the spiral model is not frequently used as its predecessors because it certainly requires hands-on experience on risk assessment and management.

2.3.2 WINWIN Spiral Model

Boehm [57] improves the spiral software process model, giving major consideration to stakeholders' communication, their goals and negotiation activities. To this end, he incorporates the spiral model with the management Theory-W. A management theory aims to make all stakeholders a winner by assuming each stakeholder pursues satisfactory agreements (i.e. win conditions) [63]. The idea of stakeholder negotiation aims at balancing between functionality and performance of software over cost and delivery times [64]. This version of the model determines goals, limitations and alternative solutions, the crucial milestones (anchor points) of software projects, *success-critical stakeholders* [14], and setting up some wining conditions for them [65].

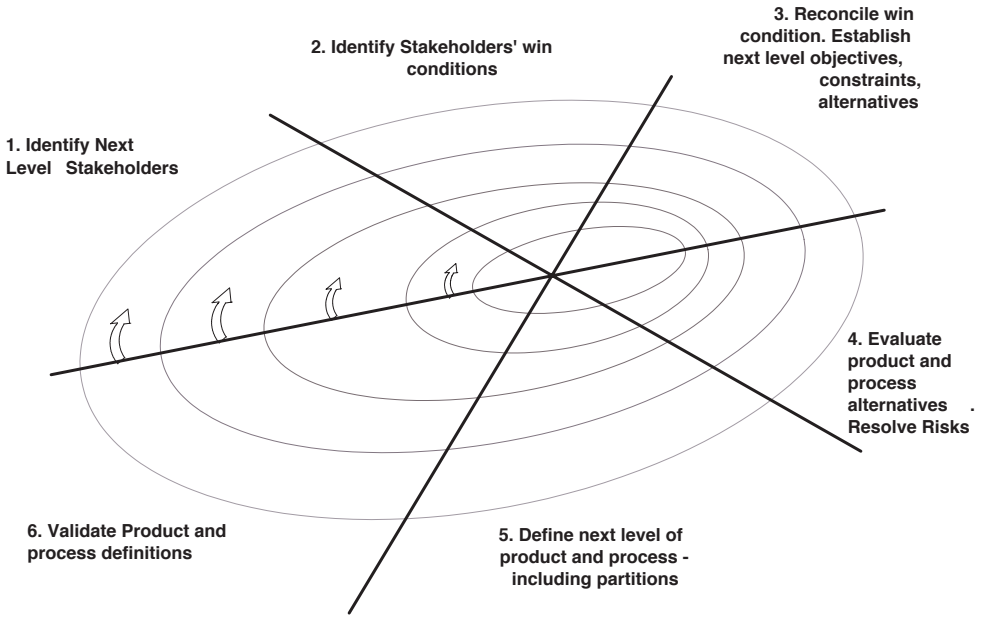


Figure 2.3: *The WINWIN spiral development model*

The WINWIN spiral model is based on a series of negotiation activities starting in each spiral. To achieve a win-win situation, multiple communication pairs among the stakeholders are configured in the following steps: (i) identification of the success-critical stakeholders, (ii) detection of the win-win cases among these stakeholders, (iii) bargaining among the stakeholders for win-win situations for revealing the satisfactory conditions, (iv) "*value-based monitoring and the control of win-win equilibrium through out the development process*" [14] (pp. 139). Furthermore, the successful results from these steps most likely produce a win-win result (see Figure 2.3 for a illustration of WINWIN Spiral Model).
The WINWIN Spiral Model suggests that the project specifications and the decision boundaries should be very flexible (e.g. based on the stakeholder decisions a software module can be bought, or alternatively a prototype may be developed). Moreover, even the process model could be changed by stakeholder negotiations (e.g. from waterfall to iterative) [62]. One important contribution of this model is that it addresses the negotiation of stakeholders and promotes cooperative teams at software organizations.

2.4 Software Process Improvement

The field of software process improvement (SPI) is established as an engineering management approach. It stems from both software engineering domain and the field of management information systems [66]. *Software process improvement* can be defined as a set of methods or bunch of activities organized to improve the efficiency of software practices [3]. This improvement is actually realized by the application of scientific techniques to observe the improvement progress in a process, services and products [67]. One of the main goals of improvement process is to understand the underlying working mechanisms of an organization (e.g. business value creation activities) and make the organization more efficient by; (i) concentrating the right activities that the organization wants to progress in (define a process), (ii) creating tools or methods to help the organizations or individuals to do these things more efficiently (asses the process), (iii) observing the ways to improve (refine the process) in a period of time while in progress [49]. The continuous improvement cycle is based on the scientific method. It was first introduced by Alhazen "*as a systematic observation of a phenomena and analysis of data relation to a theory*" [68, pp. 139]. Later, it was westernized as the Shewhart Cycle [69] that has the following states: Plan, Do, Check, and Act (PDCA). This control circle continues by amending and merging with the previous information. The notion of improvement oriented quality management was used by Deming, which was developed for the Japanese industry. Furthermore, Humphrey [41] claims that the techniques that Deming envisioned for continuous process improvement for Japanese industry are totally applicable to software environments. However, evidence suggests that unlike industrial production, the techniques of a process and a product improvement cannot that easily be applied to the software development and production processes [43]. Although, in practice, tailoring a software process is a complex task especially for a specific software project, many software development organizations report several benefits of using software process improvement and benchmarking techniques in their practices, for example, by following a guideline such as CMMI (Capability Maturity Model Integration). As introduced by Humphrey [41],

CMMI recommends a series of improvement efforts for achieving the capability of defining maturity levels. It is a model for assessing an organization's software processes for determining maturity (i.e. level of quality) of the software processes. CMMI can be enacted as a roadmap, which relies on the workable definition of process and the concept of improving organization [70]. As in manufacturing environments, a tested and well-understood process will certainly bring better quality than an uncontrolled one. In order to overcome certain software process improvement problems, CMMI suggests to form a learning organization, which will establish a landscape of a continuous improvement [71]. IDEAL is an acronym (Initiating, Diagnosing, Establishing, Acting, and Learning) for each phase of the process improvement lifecycle model [61]. As an alternative to Shewhart cycle, it has been created for assisting CMMI. The main objective for an organization is to follow the guidance (i.e. recommended practices) wherein these process areas are designated based on goal-practice structure [70]. CMMI can be used to program, clarify, execute, deploy, measure, and improve the processes in an organization.

ISO 9000 defines quality standards, which are generic and applicable to any (software) development organization as high-level quality expectations as well as to any other production environments [72]. It clearly supports the usage of the PDCA approach. ISO has developed international standards for software process assessment, called ISO/IEC 15504 (Software Process Improvement and Capability Determination) [73], which helps to define process maturity levels for processes from the very beginning level through well defined and documented stage. It is intended to be used in three different ways; (i) capability determination of software suppliers, (ii) process improvement, and (iii) self assessment of an organization for its ability to realize a software project [74, 75].

2.5 ISO/IEC 12207 Software Life-cycle Model

ISO/IEC 12207 [46] is an international standard for software lifecycle processes. Without providing details of how, it furnishes a comprehensive set of process activities and task structures for administering and engineering the activities of the life cycle of a software system. It is based on three main process groups including: (i) primary lifecycle processes, (ii) supporting lifecycle processes, and (iii) organizational lifecycle processes. The structure of ISO/IEC 12207 relies on the qualitative definitions of the processes, which are informally described (i.e. no details about the implementation of tasks or activity performance). In fact, a software development organization should select a lifecycle model to detail the implementation of these activities.

Such a model equips the stakeholders with tools, which may help them to define the purpose and outcome of a process from a functional viewpoint [3]. By its well defined terminology, this framework enables a software organization to support the organizational progress by defining their partial efforts with its predefined documentation standards, which can be considered as a common language for efficient communication among the participants of the software development process.

In addition, ISO/IEC 12207 is based on both the principles of system engineering and total of quality management [76]. A software organization is encouraged to customize the required subset of processes, which may be needed for their goals. The current version of this framework defines 43 different software processes [76] and the primal process features activity (tasks), artifacts and roles. Subsequently, it advocates the importance of task and role interactions. ISO/IEC 12207 not only aims at the improvement of the proficiencies of the stakeholders but also the software organization itself. It sets a safe ground for creating a social environment and managing the interactions among the stakeholders [3].

Processes are composed of activities, which are basically formed from set of tasks. The tasks are structures designed to process the resources into outcomes (see Figure 2.4) [76].

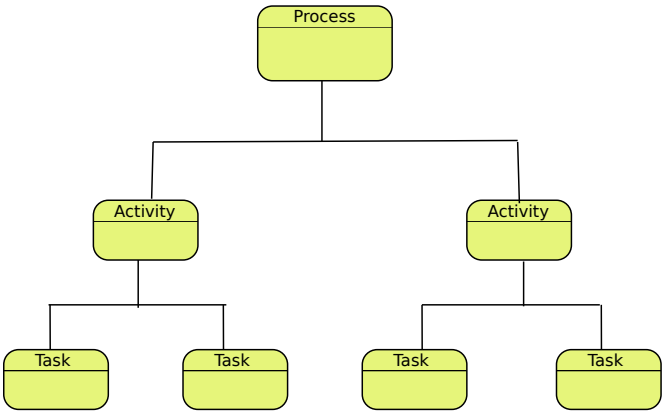


Figure 2.4: *The Structure of a Process*

2.6 Agile Methods

As the conventional software development models cannot cope with changing requirements efficiently, a group of methods called *agile methods* have emerged. They collectively promote the notion that it is unnecessary to do a detailed system design at the early stages of a project [77]. From an agile perspective, an early release helps to improve the design quality, which means testing for early defect detection and elimination. The very basic idea of agile methods is also supported by Humphrey's *Requirements Uncertainty Principle*: *A software system cannot completely be specified in terms of its requirements until it is tested by its users* [78].

Based on the fact that agile methods share the iterative development model as an underlying mechanism [79], Cockburn [80] clarifies that the aim for all agile methods is to find out (light and adequate) rules of communication and coordination to moderate the various behaviors of project.

The Agile manifesto [81] (i) prioritizes people and their interactions over tools and processes (cooperation with close communications) (ii) improves the design quality continually by the feedback from users (product flexibility), (iii) responds to a change rather than following a strict path of development with small releases and rapid production and feedback cycles (adaptability for changes), (iv) highlights the importance of working software over extensive documenta-

tion; therefore, methods should easily be learned and altered (method modifiability) [81]. In that regard, a group of researchers such as Pekka Abrahamsson, Barry Boehm named them as *light-weight* methods because they are not institutionalized with a complex form of process [82]. In addition, agile methods emphasize the importance of verbal communications and tacit knowledge in team relations, and they also aim to reduce the perceived need for extensive documentation [82]. Highsmith and Cockburn [83] claim that the novel thing about the agile philosophy is that it highlights the importance of people and their communication processes for a successful project completion. Miller [84] defines agile methods as incremental steps of methods that are (i) collaborative and people oriented for lowering the cost of communication, (ii) modular in terms of implementation, (iii) iterative with fast testing cycles with less bureaucratic activities, and (iv) adaptive for mitigating new risks.

Unlike traditional methodologies where the extent of a delivery is identified by the time needed, agile methods use *time-boxed iterations* (i.e. usually same length iterations based on scope and quality of working functionality) [85]. In particular, this means the project risks are mitigated by defining the scope in a limited time frame instead of enabling the scope to create a release length. Agile iterations encircle all phases that are required to deliver the product or a part from the product line of development.

2.7 Chapter Summary

This section provided a general discussion on the basic principles and concepts of the software development process models and methods. It started with the definition of software process and briefly explained the selected process models and the agile methodologies, and several paradigms of software process improvement. The WINWIN spiral model was the first software development model that highlighted the importance of a social equilibrium among the stakeholders from a value-based point of view. Therefore, we considered this model as an initial approach that provided the benefits of investigation of the social preferences for better software development productivity.

Starting from a philosophical viewpoint, the next chapter presents the research processes and methodologies chosen for the study, and it further details the techniques and methods used for data collection.

Part II

Research Methodology

Research Paradigm

The second part of the thesis starts with a brief overview of quantitative and qualitative research methodologies, and continues with detailing mixed method research. Next, the reasons regarding selections of mixed-methods are discussed. Not only the case study approach but also all selected approaches for the study are surveyed. Lastly, this part closes with the two procedural models of our industrial case studies.

Chapter 3

Research Design and Methodology

3.1 Introduction

The purpose of this chapter is to discuss our research philosophy with respect to the two schools of thought where we introduce the techniques and the instruments utilized in the service of our research goals. This chapter presents the research approach, design, and methodologies to address the research problem (as outlined in the first Chapter) behind them, followed by a justification of the research methodology and the adopted research method are presented. Consequently, the latter two subsections review the structural equation modeling and grounded theory. The next section introduces the research design for the first part, and the consequent section details the research design for the second part of the study.

3.2 Research Philosophy

The research philosophy is the notion of belief concerned with development of the knowledge in which data regarding the phenomenon is collected, analyzed and further processed [86]. Independently from the researcher, the positivist research paradigm considers a phenomenon is measurable by using statistical instruments such as surveys, and observable by experiments [87], while the interpretivist approach focuses on the researchers viewpoint for understanding

the social reality [88] other than seeking for generalizable truths.

Previous studies have considered both positivist and interpretivist philosophies to have an equal importance on the world of scientific research [89], and therefore they saw them complementary to one another. Not surprisingly, perhaps, quantitative research is structured on the positivist philosophy, which accepts the reality is static and observable from an individuals' viewpoint while qualitative research approaches are usually related with interpretivist school of thought, which states that there are a number of alternative interpretations of reality that accommodates the scientific knowledge itself [90].

Gallier [91, p. 149] shows the taxonomy of research methodologies with respect to their qualitative or quantitative nature as in Table 3.1.

Positivist	***Interpretivist***
Lab Experiments	Subjective/Argumentative
Field Experiments	Reviews
Surveys	Action Research
Case Studies	Case Studies
Theorem Proof	Descriptive/Interpretive
Forecasting	Future Research
Simulation	Role/ Game Playing

Table 3.1: *A Classification of Research Methodologies*

Although there are several different research methodologies used in software engineering research [92], all fall into two main categories: quantitative research (e.g. survey research, experiments, and simulations) and qualitative research (e.g. grounded theory, ethnography, action research) [93].

3.2.1 Qualitative Research

Based on the idea of seeing the human as an instrument and considering their experiences, qualitative research is a naturalistic approach that aims to investigate participants' actions and words for interpretive patterns of meaning [94]. It studies mental attitudes and social behaviors frequently recorded as information from the participants' own words and definitions, and classify them from their natural work settings or environment [95]. Qualitative researchers use techniques like interviews, individual experiences, case studies and focus groups to capture data about the values and emotions of people for investigative ob-

servations. The collected data can be documented in a contextual framework for conducting a closer observation of words and view of the participants and further inspection [96]. Typically, qualitative methods are inductive in nature; they are used to investigate a new or unexplored phenomena or sometime to generate a theory. In particular, it is beneficial in the cases where researcher needs interaction with participants to seek in-depth answers or research that requires group interactions (e.g. interactions between both respondents and researchers). However, typically qualitative studies are conducted in small groups or with a limited number of participants; hence, the results in many cases are not generalizable [97].

3.2.2 Quantitative Research

Quantitative research is the study of methods and techniques for an empirical investigation of a phenomenon by collecting and analyzing numerical data using mathematical methods [98]. The goal is to quantify the interrelations between different types of variables (e.g. independent, dependent) using statistical techniques [99]. Quantitative research is corroborative and generally based on a conventional and a systematical process to gather data, which describes the information by cause and effects relations in a rigorous way so that a number of facts and analysis results can be produced [100]. Based on the assumption that the world operates with a set of physical and natural laws, a quantitative researcher aims to test a hypothesis and sometimes conducts studies to observe the cause-and-effect relationships among variables with empirical investigations [99]. In some cases, the data required for the analysis is not available in a suitable form but is transferable to survey instruments. In a positivist approach, Verschuren [101] argues that quantification relies on the separation of social reality in terms of units and variables such as research units (e.g. software organization), observation unit (e.g. software practitioners), and analysis units (e.g. research material transformed into results). Table 3.2 indicates the differences between qualitative and quantitative approaches in terms of several characteristics adopted from VanderStoep and Johnson [90, pp. 7].

Characteristics	Quantitative	Qualitative
Data Types	*Numbers*	*Words*
Research Questions	*How many? How much?*	*How? Why?*
Data Collection Methods	*Surveys*	*Interviews, observations*
Sampling Type	*Statistical Sampling*	*Snowball or quota sampling*
Sample Size	*Large is preferred*	*Small is preferred*
Goal	*Prove/Verify*	*Discover/Explore*
Generalizability	*Generalizability is a goal*	*Generalizability is not a goal*

Table 3.2: *Qualitative versus Quantitative Research*

3.2.3 Triangulation and Mixed Methods

By using pragmatism in its philosophical underpinnings, a blending of different qualitative and quantitative approaches (or their variants) as a technique to improve the validity of research findings is called mixed method research [102]. While investigating the same phenomenon, there could be a number of advantages of using a hybrid approach. Firstly, it has the potential to investigate situations where other approaches with a specific philosophical viewpoint are constrained in a single perspective. Secondly, a mixed method strategy is useful for dealing with the weaknesses of either quantitative or qualitative methods that are employed where potentially valuable conclusions can be obtained from their proper combinations [100].

Triangulation [103] is one of the most well-known mixed method design strategies, which relies on the fact that it is vital to view a research in more than one standpoint to minimize the bias. To deal successfully with *intrinsic bias* of a single method, observer or a theory, this approach advocates that multiple types of data and methods can support hypotheses, incident or events [104]. In other words, the term triangulation is a metaphor, which describes the use of multiple methods in a single research: a combination of two research traditions, theories, data sources, methods or techniques to study a research question or to measure a single construct [105].

Tashakkori and Teddlie [106] highlight three reasons for conducting a mixed method research: (i) it allows both confirmatory and exploratory research questions to be answered by tailoring the suitable features of each methods, (ii) it enables researcher to investigate more complex phenomena in the field because

of its rich input of resources, (iii) it represents different philosophies in a single study which allows us to conduct more comprehensive empirical study.

In conclusion, a mixed method research approach can be beneficial for taking a pragmatic approach as a guiding mechanism for the research efforts. For example, during the research process, if a hypothesis is needed to be tested, quantitative methods are used. In contrast, if meaning is needed to be investigated in depth, qualitative methods are preferred. A mixed method approach can be exploited (i) to improve the analysis by using a different viewpoint, and (ii) to investigate the research problem in-depth to withstand any opposition more effectively [100].

Andrew and Halcomb [107] suggest six strategies for conducting mixed method research namely (i) sequential explanatory, which usually seeks to explain quantitative findings by using a qualitative phase, (ii) sequential exploratory design, which begins with a qualitative phase through a quantitative phase (i.e. suitable when the subject matter is not well-known), (iii) sequential transformative design, in which data collection process is guided by a previously specified theory (e.g. a type of data collection activities such as surveys are followed by a different types of data collection activities such as interviews), (iv) concurrent triangulation design, in which different types of data is collected simultaneously to approve each other, (v) concurrent nested design in which one method of data collection governs the other while both data types are collected concurrently, (vi) concurrent transformative design, which is similar to a sequential transformative design. However, in this approach, the qualitative and quantitative data are collected in parallel [107].

3.3 Choice of a Research Methodology

A research methodology is a combination of several methods, assumptions, models, techniques which constitutes the procedures for collecting and analyzing the data, measuring progress and research success in order to solve a research problem [108]. The selection of a research methodology is derived from several factors from previously conducted research and existing theories to the field, time, resources, industrial accessibility, the known and unknown variables, and lastly research goals and questions [100]. To validate the results, both numeric and textual data required by the research should be collected. The mixed-method research (i.e. triangulation) is an invaluable technique, which compares and contrasts findings and further confirm results with empirical reality using the advantages of different approaches [109].

Seaman [110, pp. 571] provides an in-depth analysis of qualitative research showing its relevance to empirical software engineering research, and she claims that "*nearly any software engineering issue is best investigated using a combination of qualitative and quantitative methods. Several scenarios are described [in this study] illustrate different ways of combining these research methods.*" Such an approach provides a comprehensive perspective about a research problem, which can be seen as a way to bridge the gap between different philosophical stances. However, the mixed method research consumes more resources and time, as well as greater budget [111].

Sequentially blending qualitative and quantitative methods, this study employs a mixed-method research strategy. In general, there are several reasons for the selection of mixed method research for this study: (i) to avoid bias problems either from a single data source or to reduce the bias introduced by the researcher, (ii) to increase the rigor of the findings by using an adequate combination of perspectives with different advantages, e.g. using methods to validate a research question from different perspectives.

In particular, we select a mixed method approach for variety of research-specific reasons, which can be grounded in the applied nature of software engineering. Firstly, we derive our research questions to formulate a contextual understand-

ing of the phenomena by direct observations from the field to support our theoretical claims. Consequently, a mixed-method approach enables us to use both the software engineering literature and our industrial contacts to construct a multi-dimensional perspective. Secondly, as we assess hard-to-measure constructs (e.g. productivity, social capital), this technique allows us to be pragmatic rather than ideological as we are conducting a research to deal with complex socio-technical issues in a complex human-centric environment. Therefore, it is crucial to follow a stepwise method: (i) to discuss our findings with the software practitioners, and (ii) to gain benefit from their industrial experiences. Thus, it is evident that a mixed-method approach is suitable for this study.

To strengthen the rigor and the validity of its results, this study used methodological triangulation with a sequential transformative design strategy. To pick several advantages obtained from both qualitative and quantitative approaches, a mixed method research was found adequate for investigating the aspects of a complex phenomena identified by our six research questions.

To build our research methodology, in this study, we use a variety of qualitative and quantitative methods including case studies, surveys, focus groups, statistical techniques such as structural equation modeling, role/game playing and quantitative analysis of our game results. In order to substantiate the credibility of our findings, we suggest a number of pairings of the research methods and utilize multiple research perspectives to interpret the results. In addition, we used semi-structured interviews, which is a data collection technique allowing freedom to follow the emergent themes in a conversational way, and expert reviews as an evaluation technique in which experts put themselves in the position of novice users to identify problems [112].

This study uses a sequential exploratory design approach for combining the theoretical findings with empirical data collected from the target population where a qualitative phase is followed by a quantitative phase. As a part of our research strategy, additionally, in-depth qualitative follow up studies such as validation interviews were also conducted after a quantitative analysis.

Figure 3.1 shows a schematic description of the overall research design. The three steps of the design can be identified as (i) requirements of this research including current context of the research, research questions, literature reviews, and the theoretical models based on these information, (ii) industrial implementation, which includes two case studies that use a mixed-method approach as discussed above, (iii) empirical evaluation as a part for both case studies in order to inspect the significance of the results. In addition, Figure 3.2 illustrates the theoretical and practical contributions, and most importantly the empirical contributions grounded in software development landscape whereas Figure 3.3 and Figure 3.4 detail the rigorous steps for conducting the case study I and case study II.

In the following part of this section, we overview selected research methods.

3.3.1 Case Study Research

A case study is a multi-dimensional activity that is frequently used for seeking answers to scientific inquiries. It is a classical approach used since the early days of scientific research usually when one or multiple instance of a phenomenon is investigated in its natural environment at a bounded time [88]. While the quality of the results and the combination of its components vary, case study is found to be an appropriate methodology for conducting empirical research in software engineering landscapes [113].

However, a basic requirement is that the researcher should study the case in-depth to collect information from individuals or organizations. According to Creswell [114], "*Case studies, in which the researcher explores a single entity or phenomenon (the case), bounded by time and activity (an event, a process, an institution, or a social group) and collects detailed information by using a variety of data-collecting procedures during a sustained period of time.*" [114, pp. 12]. To conduct a successful case study research, the site that the case study is conducted should be accessible and the key people and the required resources need to be available [115]. Ultimately, a case study does not require a researcher to control over the events and situation as in action research [116].

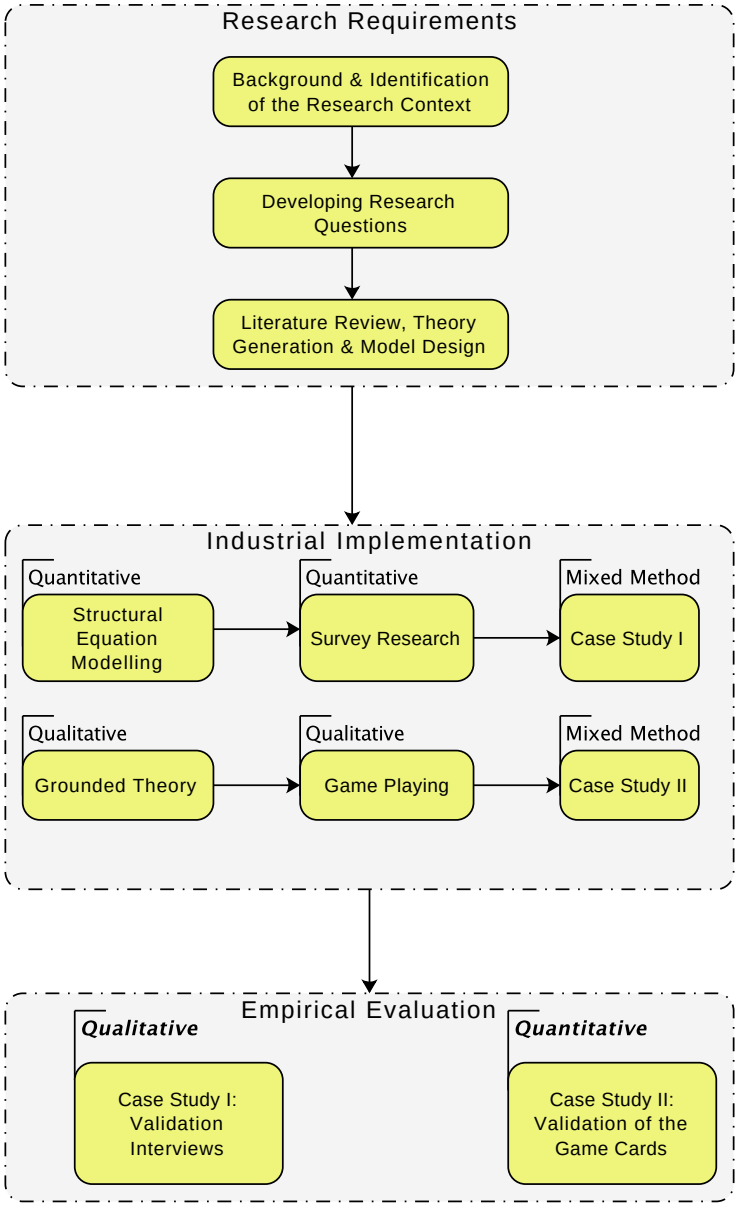


Figure 3.1: *Overall Research Design*

Yin [117] suggests that a case study should be *an empirical inquiry*, which examines a state-of-the-art phenomenon in the real-life situations based on multiple information resources with unclear boundaries in the given context. In his discussions about the meaning of a case study as a research strategy, Verschuren argues: "*A case study is a [triangulated] research strategy that can be qualified as holistic in nature, following an iterative-parallel way of preceding, looking at*

only a few strategically selected cases, observed in their natural context in an open-ended way, explicitly avoiding (all variants of) tunnel vision, making use of analytical comparison of cases or sub-cases, and aimed at description and explanation of complex and entangled group attributes, patterns, structures or processes." [101, pp. 137]. Yin [117] points out six different sources of data (evidence) suitable for case studies; (i) documentation, (ii) archival records, (iii) interviews (or surveys), (iv) direct observation, (v) participant observation, (vi) physical artifacts. They can be used as a single source or complementary to one another. Table 3.3 illustrates the sources of data, their strengths and weaknesses adopted from Yin [117, pp. 86] .

Source of Evidence	**Strengths**	**Weaknesses**
Documentation	• stable - repeated review • unobtrusive - exist prior to case study • exact - names, etc. • broad coverage - extended timespan	• retrievability difficult • biased selectivity • reporting bias reflects author bias • access may be blocked
Archival Records	• same as above • precise and quantitative	• same as above • privacy might inhibit access
Interviews & Surveys	• targeted - focuses on case study topic • insightful - provides perceived causal inferences	• bias due to poor questions response bias • incomplete recollection • reflexivity - interviewee expresses what interviewer wants to hear
Direct Observation	• reality - covers events in real time • contextual - covers event context	• time-consuming • selectivity - might miss facts • reflexivity - observer's presence might cause change • cost - observers need time
Participant Observation	• same as above • insightful into interpersonal behavior	• same as above • bias due to investigator's actions
Physical Artifacts	• insightful into cultural features • insightful into technical operations	• selectivity • availability

Table 3.3: *Six Sources of Evidence for the Data Collection*

A typical case study can be formed from a single case or from multiple cases both of which can be based on a holistic (single) unit or embedded (multiple) units of analysis, and therefore four kinds of case study designs are available: (i) *single-case embedded*, (ii) *single-case holistic,* (iii) *multiple-case embedded*, and (iv) *multiple-case holistic* [117]. The case study is usually constructed as a tool to connect collected data with initial research inquiry, which allows the researcher to draw a set of conclusions.

According to Kitchenham [118], the usage of case studies has several advantages: (i) they can be combined with software engineering activities, (ii) if real projects are used, there is no need to increase the size because they are already on the actual industrial scale, (iii) they enable the researcher to asses the actualized and expected benefits of the progress. Basically, there are four main steps in a case study: (i) design, (ii) conduct, (iii) analyze, and (iv) draw conclusions. Finally, industrial case studies are quite important for appraisal of software engineering instruments and processes to cope with scale-up issues, eliminate biases, and ensure validity (e.g. internal, external, etc.) from an evolutionary perspective [119].

3.3.1.1 Threats to Validity

In scientific research, the potential factors that adversely affect the accuracy, usefulness and quality of results are called *threats to validity* [120]. These threats, however, should be addressed in every step of a case study where they can be classified in four dimensions of validity with the criteria to judge design quality of a case study based on (i) construct validity: *correct operational measures and constructs should represent the subject matter*, which should also interpreted similarly both by the researcher and the participant [113]; (ii) internal validity: *the conceptual definitions should match operationalization as it was predicted* where there may be other invisible factors affecting the validity [117], (iii) external validity: *the study should provide an extrapolation of the results beyond the initial study*, in which researcher investigates the relevance of the findings for similar settings and other people [121], (iv) reliability: *is the sta-*

bility of the measuring instrument with which the study should be repeatable with the actual results [117]. Therefore, the way the research is conducted and the protocols are followed should be defined precisely. Table 3.4 summarizes potential threats to validity for the study [113, 120, 121].

Construct	*Ability of an instrument (e.g. survey, test, scale) to measure a concept properly.*
Validity	• Qualitatively, check if the researcher and the participants have the same interpretation over the results, e.g. validation interviews. • Quantitatively, check measures of a construct, e.g. assessing a model with confirmatory factor analysis.
Internal	*The design of the research should have internal coherence, strength, and soundness.*
Validity	• **History effect**: An outside situation or a factor may cause a manipulation over the results. • **Testing effect**: Continuous testing may lead to experimental deterioration. • **Instrumentation effect**: Any change in the measuring device during the study. • **Experimenter or participants effect**: The findings of the study can be biased by participants who aim to assist the researcher and change their preferences as a result.
External	*The results of a study could be extrapolated and should be able to represent the target population.*
Validity	• **A conceptual replication:** An alternative study based on an initial study uses either different methods or measures to test the same constructs differently. • **A systematic replication:** Rigorously change a parameter to observe findings by alternating a setting or changing a group of the participants. • **College sophomore problem:** Using university students for empirical studies instead of industrial practitioners.
Reliability	*The measuring instrument should be precise and stable where alternative researchers could be able to reuse the measurements, methodologies and data collection protocols.*

Table 3.4: *Threats to Validity for Empirical Research in Software Engineering*

3.3.2 Survey Research

Survey research method uses questionnaires to collect data in a systematic way from a group of individuals such as organizations, teams, and students [88]. There are two approaches in conducting surveys: (i) a cross-section approach which gathers data at a single point in time, (ii) longitudinal survey research which repeats the observation of the data over a period of time [122]. However, a result of a survey demonstrates an association but not a causality (i.e. a causal link between variables) [118]. In addition, the results may be considered as biased if the interrelationships between the population and the number of respondents are not well known [123]. Although there are no limitations, some

researchers believe that a survey research is a quantitative approach [88]. A survey is a measuring instrument. The responses of a survey represents the existing opinions of participants on a subject matter, which can be used to draw conclusions. The questions employed in a survey should be simple, clean and written without a jargon [99]. To maximize the validity of the results of a survey, a researcher needs to construct a strong theory, employ a good survey design and utilize proper statistical tools [124].

3.3.3 Structural Equation Modeling

A family of flexible interrelated statistical techniques (i.e. multivariate, multiple regression analysis, factor analysis) frequently used in social science studies to analyze empirical data and test variables and evaluate their network of hypothesized relationships is called structural equation modeling (SEM) [125]. Based on the patterns of statistical expectation, it is a confirmatory multivariate analysis technique used to estimate the structural or casual relationship among two variable types (i.e. observed and latent). SEM models use a collection of simultaneous equations based on a combination of observed and latent variables (hypothetical constructs or factors), which are frequently used by sociology, psychology research and econometric research [126]. The main component of a structural equation model is an initial hypothesis, which also includes the components that may be connected that are assessed by several statistical tests and if necessary adjusted through modification indexes.

SEM allows the researcher to explore the multivariate relationships that can be used to test an actual hypothesis, which may theoretically be justifiable by empirical observations. A typical SEM model usually encompasses the graphical depiction of the correlation patterns based on a set of variables, and is frequently used for validation of the relationships among the latent constructs. Although it is a quantitative approach, SEM offers a start from a qualitative viewpoint; it has the ability to show how the chosen factors or variables are not only correlated but also interrelated to one other. Therefore, it can be helpful for observing the relationship among several coefficients. It enables the

researcher to assess the effectiveness of a hypothetical model for the sampled data. In particular, a model based on the combination of regression, path, and confirmatory factor analysis should be useful for analyzing social factors and their interdependencies.

In addition, it is sometimes used as an instrument to form a measurement scale. A typical SEM includes the direct and the indirect associations of variables that are statistically assessed to identify a relationship between data and the proposed or hypothetical model. Consequently, the notion of correlation and covariance is important for a SEM analysis because they signify the pairs of relationships for a group of variables [127]. Correlation is a tool that defines the discovered linear relationship between two variables (coefficient of correlation measured in a range of -1 to +1). A positive value indicates that there is a positive correlation among the variables, where negative values state the opposite [128]. In fact, SEM is considered as a set of equations used to compute a multiple linear regression model where several factors are calculated with respect to observed weights [129]. A SEM model can be used for measuring the correlations and covariation among the latent constructs, *where the regression model is designated in the structural part of the model, and factor analysis model is designated in the measurement model* [130, p. 10]. There are four main steps in a typical SEM analysis; (i) model development (building a conceptual framework), (ii) path diagram construction (building a representation of associations), (iii) assessment of measurement model, (iv) assessment of structural model [131].

A SEM can be specified in several formats such as path diagrams. However, these figures usually follow *de facto* standards. A typical SEM model represents how the researcher relates the hypothetical constructs and the collected data based on observed variables (illustrated in rectangles). These variables are derived from a set of questionnaire in a survey tool. To represent these items, a limited number of graphics are used such as ellipse, which signifies the latent constructs that are estimated from the observed variables, single headed arrows, which represent predictive relationships and a double headed curved

arrow between two latent variables, which indicates that they are correlated. Based on the variance-covariance matrix, a good-fitting model designates that a theoretical or hypothetical construct is consistent with the empirical dataset. Such a model is useful for examining the relationships of the causal paths of a SEM model, which can improve the original form [132]. However, sometimes a model that seems like a good-fitting model may not be a working model. Therefore, it is important to use several model validation techniques to evaluate the validity of a SEM model to obtain more conclusive results. A chi-square test, the comparative fit index (CFI), the goodness of fit index (GFI), the adjusted goodness of fit index (AGFI), and the root mean square error of approximation (RMSEA) are the most common fit-indices used in SEM investigations [126]. In addition, sample size is another parameter that affects the validity of a model [126, 130], where a number of researchers suggest that constructing a model with no latent variable is somehow more suitable for a limited sample size.

One of the earliest current fit-indices in SEM research is the chi-square test statistics. It is frequently used for testing the model fit by investigating whether a null hypothesis is true or false. Barrett [133] argues that a chi-square test is enough for investigating the model fit. Although for a large sample of data this test usually shows statistically significant results, it is still used as a measure of general model fit to identify whether a theoretical model differs from the sample variance-covariance matrices calculated from the data [129]. It is affected by the highness of the correlations, which results in poor fit for the proposed model. Moreover, the evidence collected from simulation studies confirms the sensibility of chi-square test in terms of the size of the sample set [134].

The root mean square error of approximation (RMSEA) index is probably the best-known index for model fitting. Analogous to other fit indices, RMSEA uses a complexity parameter depending upon the degrees of freedom of a model [132]. According to Browne and Cudeck [135], RMSEA value measured below .05 indicates a good model fit between the observed data and theoretical model, while values below .08 is considered as a reasonable fit [131].

Based on the parameters identified above, we select a set of indices to evaluate the models constructed in this study, namely chi-square goodness-of-fit test, ratio of chi-square to degrees of freedom, root mean squared error of approximation (RMSEA), and two other kind of measures known as goodness-of-fit index (GFI), and adjusted goodness-of-fit index (AGFI). Table 3.5 presents the descriptions of and thresholds for several indices based on the works of Bagozzi and Yi [136], Cote et al. [137], and Ping [138], etc.

Fit index	*Descriptions*	*Cut-offs*
χ^2	*Displays the disagreement between hypothetical model and collected data*	$p < .05$
χ^2/df	*As chi-square test is depended on the size of a sample*	2-1 or 3-1
RMSEA	*Displays the level of fitness of a model*	<.05 good <.08 reasonable
GFI	*A de facto measure of the descriptive adequacy of a model*	0 no-fit, 1 perfect-fit
AGFI	*GFI adapted for degrees of freedom*	0 no-fit, 1 perfect-fit
NNFI	*Displays the level of improvement compared to null model*	0 no-fit, 1 perfect-fit
CFI	*Shows betterness of a model fit with respect to a null model*	0 no-fit, 1 perfect-fit

Table 3.5: *Descriptions and Cut-offs for the Fit Indexes Adapted from [131]*

Lastly, so as to apply SEM properly, the hypothesized measurement model should be illustrated by a diagram in which measured (observed) variables are called factors or indicators. In a SEM model, observed variables are represented in the form of rectangles where latent (unobserved) variables are shown by a circle and the relationships between these variables are usually shown by arrows. To achieve a precise measurement result, the indicators that are used to measure the latent constructs should be validated by using methods such as literature reviews, and expert reviews.

3.3.4 Game Playing as a Data Collection Method

In this subsection, we introduce a novel, game-based approach as a part of our research method where data is collected by conducting a card game. One of the goals of these card-based assessments is to understand the relationship between participants' perception on an object (e.g. cards, images, etc.) and subjects' patterns of behavior or sometimes their personality traits.

There are several techniques in psychology used to construct a game-based approach, which are mostly categorized as *projective techniques* [139]. For example, the *associative approach* requests the subject to respond to certain cues such as cards or words by uttering the first thought that comes to mind. Secondly, a *constructive approach* requires the subject to complete a task such as creating a story from the objects shown. Thirdly, in the *completion technique*, the subject is requested to finish an incomplete statement or a sentence. Fourth, in the ordering or *sorting technique*, the participant is requested to do a sorting or ordering of objects, cards, pictures, etc. Finally, in the *expressive approach*, the participant is asked to express himself or herself freely.

A well-known variant of these tests uses the inkblot technique in which a card is shown to the subject one at a time and the subject describes what these cards remind him or her. In modern psychology, similar tests are also used to understand participants' social behaviors and personality types. These tests, for example, may analyze whether answers of a participant are defensive or argumentative.

3.3.4.1 Our Novel Approach

Depending upon the complexity of tasks and human interactions, in software engineering settings, a precise computational model for profiling software practitioners is hard to construct. In light of this remark, we propose to gamify the personality profiling akin to the psychiatric tests. Similar to the psychiatric assessments or card-based tests used in the field of psychology, in our approach participants are to answer a set of questions. However, our technique is based

on *situational context cards*[1] that are equipped with psychometric questions, which are derived from several situations captured from the events observed in the software industry. Grounded on software development concepts, these context cards are used to operationalize our game-based approach where the goal of the game is to reveal personality types of software practitioners. In addition, we suggest a systematic approach not only for revealing the personality profile of an individual or a software team but also understanding the personality profiles of organizations as a whole.

To this end, we provide a new understanding of collecting data: game playing as a data collection method. Instead of collecting paper-based data for a psychometric assessment, we propose a game-based assessment with more interactive questions for the data collection process. Where we have termed this novel approach as *qualitative simulation: a scenario based information gathering, analyzing, and evaluation method, which relies on a card game component with a deck of cards to play for the results.*

We termed the concept as qualitative simulation for two reasons. First, we design a game where game like instruments are generally perceived as *qualitative* in their nature. Secondly, similar to a simulation, based on real life and context dependent situations, our approach is likely to operationalize scenarios and events that can happen during the software development life cycle.

Consequently, we construct a game like approach, which is designed for face to face interactions. During these interactions, participants' responses are recorded. Based on practitioners reactions to those of events, later these recordings are analyzed, and interpreted. Furthermore, the researcher plans to use such an understanding to employ hypothetical events to observe the participants' verbal behaviors, and to discover the personality structure of the software teams.

[1]Context dependent cards that are designed to store real-life situations.

3.3.5 Grounded Theory

To bridge the gap between obtained empirical data and a theory at a conceptual level, grounded theory was introduced by Glaser and Strauss [140]. It is a systematic approach for collecting and analyzing qualitative data and present them using the theory of *symbolic interactionism*, i.e. an individual's definition of a situation that causes an action [89].
Because of its integrated and iterative nature, grounded theory is an approach that requires continuous *interplay* between analysis and the data until a theory emerges [140].
According to Glaser,

> *"The goal of grounded theory is to [iteratively] generate a theory that accounts for a pattern of behavior which is relevant and problematic for those involved. The goal is not voluminous description, nor clever verification." [141, pp. 93]*

Strauss and Corbin [142] claim that it is easier to understand a phenomenon when a theory is derived substantially from socially observed situations (i.e. data) *pari passu* within the theory in an experiential world. Therefore, grounded theory allows for exploration of meaning and creates connections to analyze the understandings of reality for participants in a domain. Grounded theory acquires constructs by actualizing categories from the raw data by using a technique called *constant comparison* [140]. In a grounded theory study, the theoretical concept is developed and interrelated by breaking the transcripts (from the interviews) or field notes into categories of meanings from texts, which are found important by the participants [142], and further by comparing and contrasting the analysis, so data collection phases are iterative and continue until a *theoretical saturation* is attained [140].
Three phases are defined as *coding* in the process: (i) open coding for building information categories, (ii) axial coding for connecting subcategories with categories, and (iii) selective coding for refining the categories around a cen-

tral category by building a storyline [143]. In addition, a common technique used in grounded theory research is called memoing [144], which is an act of recording thoughts in the form of statements or questions that can be used as a supplementary material to assist the researcher in building abstractions, that is generating categories from the raw data.
To sum up, grounded theory is a methodology that promotes the idea of continuous engagement with the empirical findings while segmenting the key thoughts among the conceptual elements by possible operationalization of coding, comparing and memoing of the raw data. Furthermore, grounded theory has been used successfully in software engineering research (e.g. see [145, 146]).

3.3.5.1 Justification for Using Grounded Theory

We built the card creation process based on grounded theory for a number of reasons: (i) to our knowledge, no study exists in the literature that offers an inductive approach and therefore allows for the emergence of a psychometric structure to emerge based on the experiences (or situations) captured from the individuals in a software development organization, (ii) the grounded theory analysis is a well-established research methodology, which has sophisticated guidelines for conducting an empirical research especially beneficial in the fields such as software engineering partially based on the exchange tacit knowledge which requires social interaction, (iii) based on the industrial experience of the researchers, grounded theory promotes the notion of *theoretical sensitivity*, which relates the ability of researchers to understand the important elements of the data and its empirical contributions to theory generation (i.e. card creation) process. Additionally, during the card creation process, we believe that three-step systematic coding proposed by Strauss and Corbin [142] is helpful to figure out what data should be collected during our iterative coding approach.

3.3.6 Focus Group

Focus group is a form of group interview (i.e. researcher-led group discussion) conducted to capture a content in the research process where participants are

asked about their opinions, understandings, stories or perceptions as regards a previously selected subject [147]. It can be tailored in many ways and for different goals in which data collection can be done using group discussions. This technique is quite commonly used in the field of psychology to capture the different perspectives from a variety of participants [148]. As another example, in the field of market research, a focus group is used for developing the contents of a survey instrument, [88], for building a hypothesis, or developing a construct as an initial step before questionnaire development [147], and in usability research it can be used as a complimentary technique for evaluating the user interface of a product [149].

Typically, focus group studies are conducted using a group of participants to obtain a broad range of data in a limited amount of time. In a focus group setting, the researcher stimulates the group conversation by posing a set of questions regarding to the core topic of interest so that individuals can freely discuss about a subject. [88]. According to Howitt and Cramer, a focus group can be useful for discussions especially in the early stages of a research as it generates a mixed viewpoint, and is conductive to brainstorming about novel concepts [150]. Lastly, a focus group has "*the process that began with asking the participants to focus on the topics that were most important to the researchers ends with the researchers focusing on the topics that were most important to the participants.*" [88, pp. 354]

In addition, the focus group method has been previously conducted in the software engineering domain (e.g. see [151, 152]).

3.3.7 Summary

This thesis follows two case-study designs, with in-depth analysis of (i) factors affecting software productivity, (ii) personality traits of the teams of software practitioners. Figure 3.2 illustrates a conceptual overview of the research where three phases can be observed; (i) Theoretical Contributions: Model Building, (ii) Empirical Contributions: Grounded on the Software Development Landscapes, (iii) Model Testing and Empirical Validations.

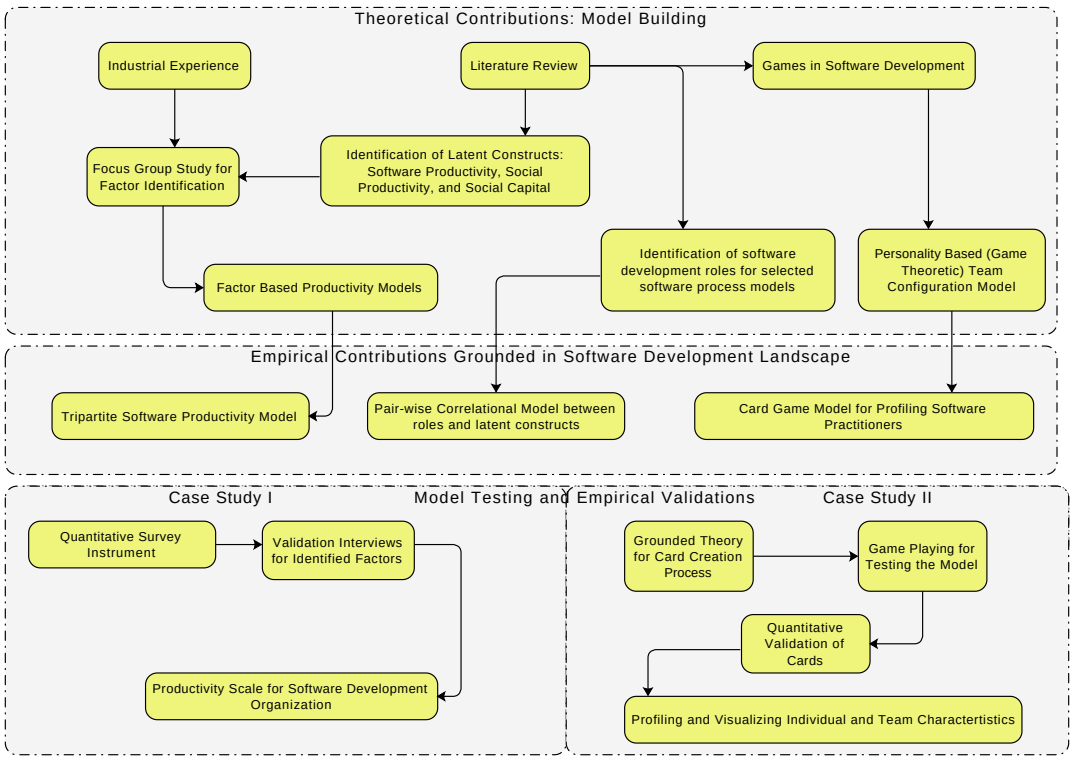


Figure 3.2: *A Conceptual Overview of the Research*

The research activities in this work can, thus, be summarized as follows: Firstly, we identify the factors that are affecting the software productivity and its relationships with its social aspects namely social productivity and social capital. Based on the selected constructs and the identified factors, we build several factor-based productivity models for software development. To assess the validity of a measurement scale based on the quantification of these latent constructs, we conduct empirical observations in an industrial setting, where we constitute a tripartite factor-based productivity model specific to a software development organization. Secondly, we identify software development roles by using a group of software process models. By using this as the main reference, we empirically investigate the pair-wise correlation between roles that are found in software industry and latent constructs. In an attempt to understand the characteristics of a personality-based team configurations, thirdly, we investigate personality traits of the software practitioners. To this end, we design

a card game, which can be used to reveal the personality traits of software practitioners on a psychometric scale.

3.4 Research Process: Case Study I

In this section, we elaborate the research methodology for the case study I (see Figure 3.2). We describe the details of the research process used for the empirical investigation of the factors affecting software development productivity by a case study analysis using a middle-sized software company. The details of our research methodology (see Figure 3.3) comprises the following steps:

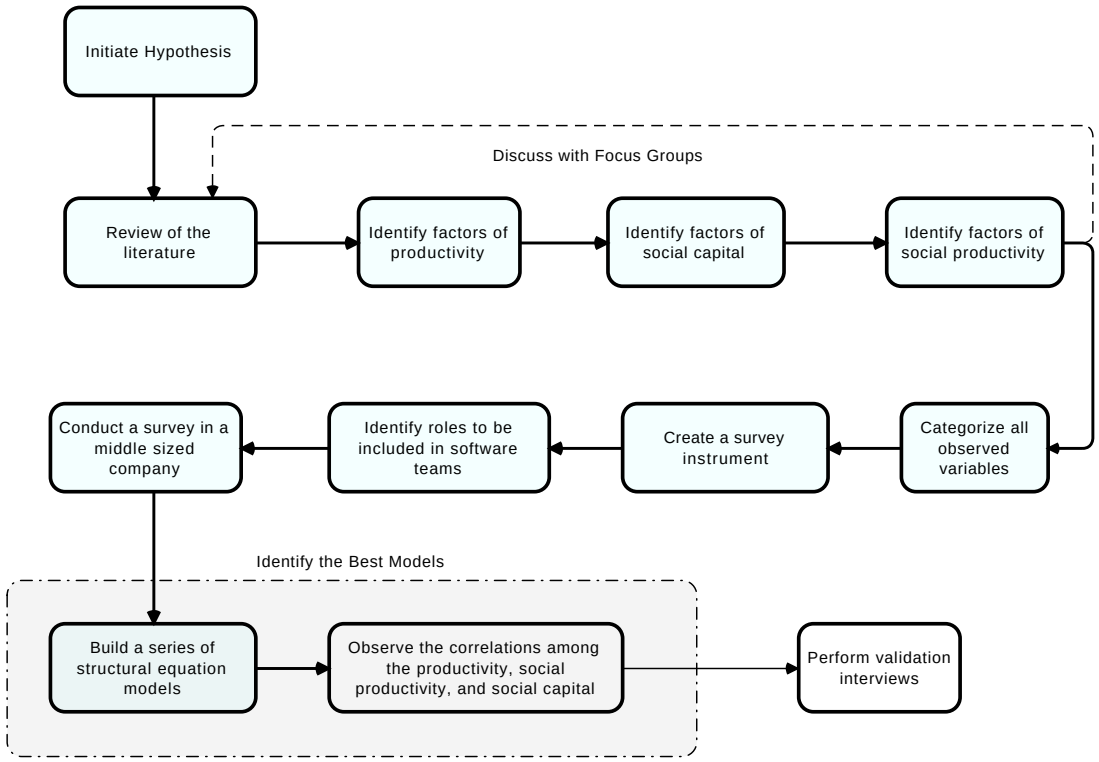


Figure 3.3: *The Systematic Approach for Creating SEM Models of Productivity*

- First, we consider productivity, social productivity and social capital as latent variables that cannot be directly observed. Therefore, we use several potential factors to identify them.

- We form our hypothesis, which represents the fact that there is an observable relationship among the pairs of productivity, social productivity and social capital based on the selected determinants.
- As identified above, secondly, we review the literature to consolidate the key factors affecting the productivity, social productivity and social capital of a software development organization.
- Thirdly, we categorize these variables for creating several models for each latent variable that are identified and for assessing their correlations.
- To investigate the degree of participant's agreement, fourthly, we developed a survey instrument with sixty questions on a Likert scale between 1 (strongly disagree) and 5 (strongly agree).
- Next, the questions are transformed into a questionnaire where the data is collected and initially analyzed. The main portion of the data is used for measuring the correlations of the factors affecting our three latent variables. We asked a set of questions exploring such dimensions as work experience of a participant, gender, actual and the ideal team size (e.g. see Appendix A for questions 18 and 19).
- Fifthly, we illustrate the framework by a case study as confirmatory factors analysis and construct several structural equation models with single, double, and tripartite models in an attempt to find the best model that fits the data collected for those purposes.
- For the models with one latent variable, we develop our first model with the data from the literature and an alternative model, which is usually refined by focus group study where we asked the company for their opinion about the determinants of selected models.
- We build alternative SEM models for each single latent construct to compare a version from a literature and a version developed using the company selected indicators from an industrial perspective (see Model II, Model IV, Model VI).

- After constructing the initial models, we investigate the impact of roles and team based parameters on the latent constructs of productivity, social productivity and social capital.
- Finally, we perform a set of validation interviews to discuss the results obtained from SEM models with the management team of our industrial partner, which yields some interesting insights and interpretations.

3.5 Research Process: Case Study II

In this section, we detail the research process we conducted in the second case study (Figure 3.2). The details of the research process (see Figure 3.4) comprises the following steps:

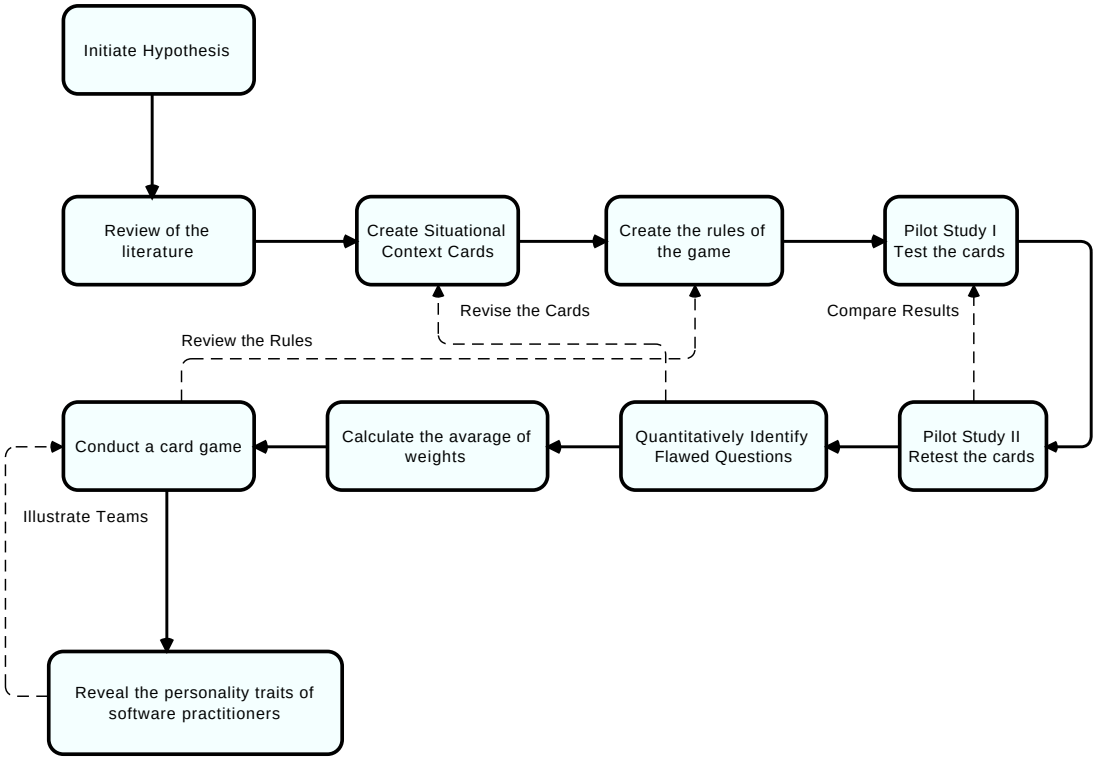


Figure 3.4: *The Steps Involved in Our Systematic Research Process*

- First, we form our hypothesis that there is a visible relationship between the effective software teams and specific patterns of their personality traits.
- Secondly, we review the literature relating to the use of MBTI tests in software engineering domain and investigate previous findings. Next, we review the coding in the grounded theory analysis, which will be based on the context cards creation process.
- Thirdly, we create the situational context cards (the card creation process is detailed in the following chapters).
- Fourth, we establish the rules of the game, which outlines the constraints for the game-based personality analysis.
- To test the situational context cards, fifth, we conduct the first pilot study and collect data for the questionnaire.
- Next, we conduct a second pilot study with exactly the same group of people to recollect the data for each question.
- In the seventh step, we use a quantitative analysis technique to single out the questions which are found problematic using both pilot tests. We perform a two-step industrial case study.
- In the eighth step, we conduct a survey on 216 participants from the same software company to investigate the importance of the questions. In light of this data, we calculate the average weights for each factor affecting the personality traits of individuals, which is used for assigning weights for each question.
- In ninth step, we conduct our game repeatedly under the same conditions with sixty software practitioners working on a number of teams.
- Finally, based on our findings on this case study, we illustrate five software team structures with respect to the personality traits of its members.

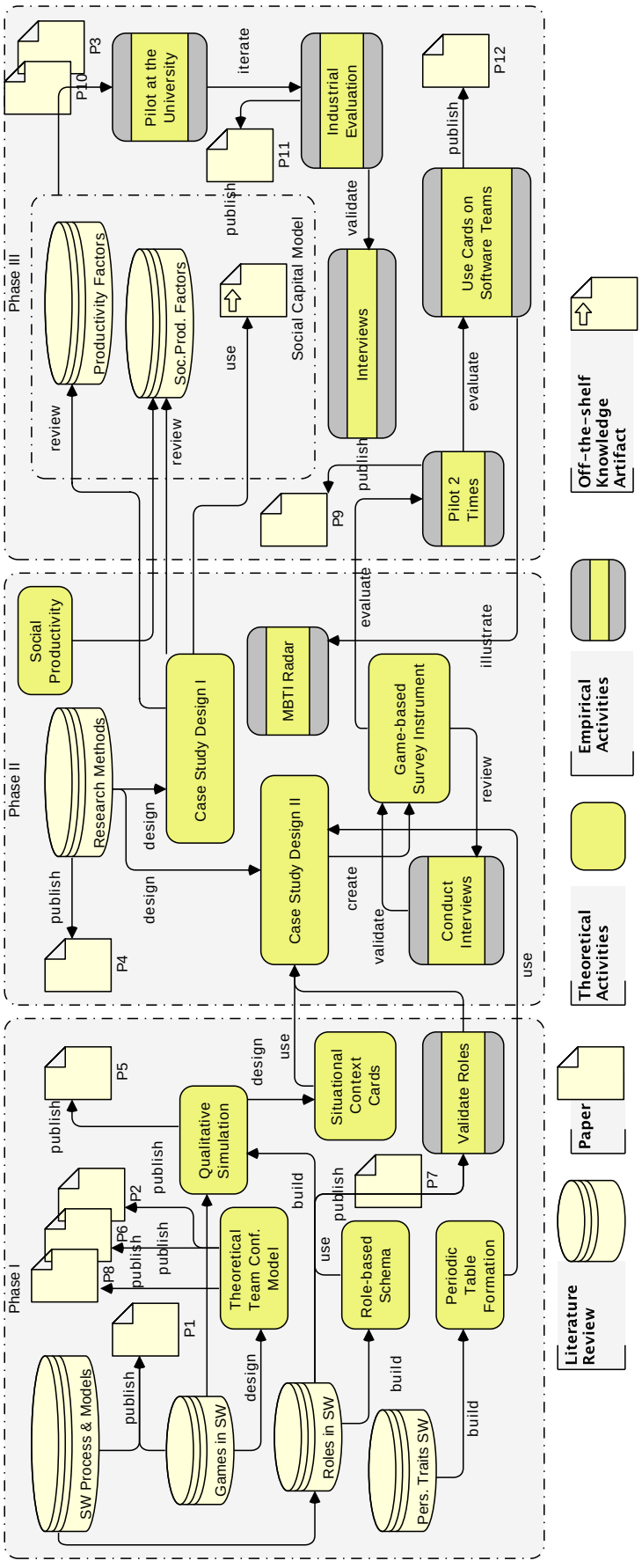


Figure 3.5: *Holistic View of Research Activities*

3.6 Holistic View of Research Activities

Figure 3.5 illustrates the holistic view of how the research papers, theoretical and empirical findings are connected in overall research design. It is divided into three main phases, corresponding to the three years of progress. To empirically test the theoretical models, a significant amount of this study was presented and published in academic conferences and journals (see page xviii for more information).

The initial research idea was published as early as possible to minimize the implementation risks (see P1). Using an iterative strategy, in phase 1, several different game theoretic models were built and the research idea was rigorously refined. The outcomes of this process were published in a number of software engineering conferences (see P2, P6, P8). To assist in tailoring the roles to software practitioners, a summary of roles in different approaches was reviewed and its results were published (see P7). The vision of using a card game to assess software practitioners was first formed and published in the Software Engineering Notes (see P5). Later, it was used in designing our second case study.

In the beginning of phase 2, to discuss the initial research plans, and preliminary research questions, a position paper is published (see P4). Based on the feedback and our theoretical models two industrial case studies were designed. In case study 1, the relationship between software productivity and social capital of software development was investigated. Concurrently, the situational context cards were created, and several interviews were conducted. The two pilot studies were performed to evaluate our cards at a university setting, and the results of these 2 pilot studies were published in a conference (see P9). An empirical study to test our game based instrument on software teams was conducted, and its results were submitted to a journal (see P12).

In phase 3, the factors of software productivity were reviewed and the social productivity of software development was introduced. An initial survey for investigating the factors of productivity was conducted as a pilot study at a university environment, and was published (see P3). An extension of this paper

was invited to a journal, which was published (see P10). Using this experience, a new survey instrument was built. Ultimately, the survey instrument was tested in the field, and results were submitted to a journal as P11.

3.7 Chapter Summary

This chapter started with an introduction to research philosophies, which was followed by a brief discussion of the qualitative, the quantitative, and the mixed method research. It continued with a justification of the selection of the mixed method research strategies. Furthermore, we discussed the set of methods that are specifically selected for this study including case study, survey research, structural equation modeling, game playing, grounded theory, and focus group. Finally, the two industrial case study approaches specifically designed as the research process were presented. In the following part of this thesis, we will detail the research processes and its first chapter is going to present the empirical analyses and their findings from the two industrial case studies.

Part III

Literature Review & Theoretical Contributions

Introduction

This part of the thesis draws its strength from several fields of research, and collates the knowledge from various disciplines including but not limited to software engineering economics, game theory, software productivity, and personality type research in psychology. It includes three chapters: the literature review of a number of academic fields and our theoretical contributions (i.e. the models that are created based on these reviews), and consequently a number of theoretical contributions that comprise the productivity and software team configuration models are built. In the following three chapters, the foundation for this research is built and documented. Among other minor contributions, it makes two key contributions: (i) building a game-based model for understanding effective software team configurations, (ii) constructing models of software development productivity. In addition, several constructs such as social productivity and social capital for software development are introduced, and a number of theoretical models have been built to reveal the relationships among these constructs.

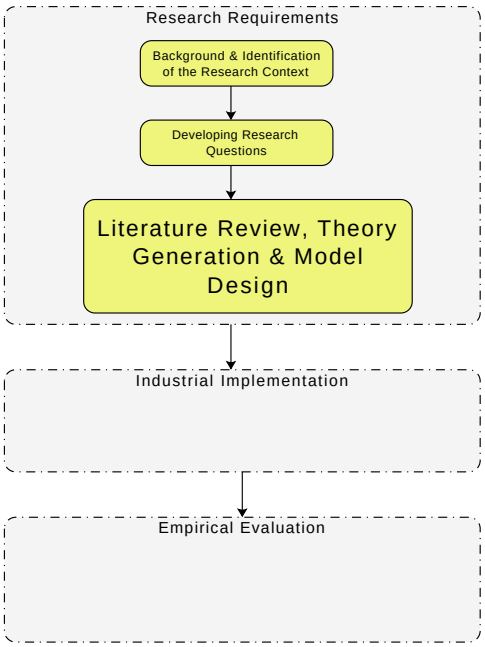


Figure 3.6: *A part of the conceptual overview of the research*

Chapter 4

Application of Games in Software Engineering

4.1 Introduction

In this chapter, the theory of games and its application in software engineering domain are investigated in detail. It starts with an introduction to games and their business applications. Then, it presents an introduction to game theory, the notion of mechanism design, and the application of games in software engineering research. The last part of this chapter concludes with a game theoretic model based on revealing participants' personality types with the goal of understanding effective software team configurations.

4.2 Defining Games and Gamification

Although differences of opinion still exist, there appears to be some agreement that a game refers to "*a system in which players engage in an artificial conflict, defined by rules, that results in a quantifiable outcome.*" [153, pp. 80]. A typical game consists of players, rules of interactions for regulating participants' behaviors, a protocol or a game system (e.g. game board) with which a player interacts, and ultimately a goal commonly based on an artificial conflict or an outcome. The notion of games is ubiquitous and highly connected with char-

acterization of human activities with several benefits [154]. From the outset, games enable us to form collective social structures, which ultimately produce considerable advantage for building complex software artifacts. Therefore, it is not surprising to discover that games that are played in societies have great (social and economic) benefits [155]. In some cases, they could even offer solutions for society based problems [35].

Most broadly conceived, the vision of using game design elements and game based thinking to create player like motivated behavior (e.g. competition, collaboration, etc.) is termed as *gamification* [156]. The trend emerges from the idea to use game mechanics and game design rules unexpectedly outside the video game industry. Considering the size of its industry, games demonstrate themselves as useful tools for incentivizing participants for a planned duration [157]. Gamification can be used to transform a monotonous process to a preferred activity by engaging and motivating its users, e.g. see [158]. In addition, a number of firms have already used gamification to improve their customer loyalty programs mostly for creating competitive advantages in their business where the application of game mechanics is one of the convenient ways to encourage customers' (e.g. by providing reputation points, achievement badges, etc.) to use their products and services [154].

However, the term gamification is not yet precisely defined. Zichermann and Cunningham [156] point out that gamification may have diverse meanings for different individuals. Deterding et at. [159, pp. 2] define gamification as: "*use of game design elements in non-game contexts*" [157]. From a marketing viewpoint, Huotari and Hamari [160] identify gamification as an activity in which quality of services are improved by using game-like features, which could potentially be useful for marketing services. This view is not fully supported by Deterding et at. [161] who argue that gamified applications should only employ *elements* of a game unlike a serious game, which eventually proposes a complete game construction. A potential limitation with this explanation is that first there is no consensus for the exact definitions of game elements [157]. Secondly, it seems that the understanding of the boundary between the game and

game element is questionable. For example, a game consists of self-similar elements, which are constructed by using a set of gamified parts; therefore, game itself may be an end-product of a gamification process. Thirdly, this researcher argues that initial descriptions fail to take process thinking into account. In fact, one possible outcome of a gamification process might likely to be a game itself. Even in some cases, a game can be gamified in the actual process (e.g. see [162]). Lastly, Groh [163] discusses the challenges and strategies for facilitating and promoting gamification where he highlights the fact that the concept should be more utilized for conducting rigorous research regarding the pros and cons. In the context of this thesis;

> ***Gamification is a transformation process in which interaction patterns, game mechanisms, reusable game components are operationalized to solve problems in an intended environment that is situated within a real world context.***

To produce a gamified system or a component, players' interaction patterns should be identified and a game mechanism (i.e. rules of interaction) should be formed. As a part of the process, the game mechanics (e.g. levels, badges, etc.) should be tailored based on the requested level of engagement with the user.
The idea of using the gamification process is relatively new in software development landscapes. From a software engineering standpoint, a game-based framework may be useful for exploring managerial problems such as understanding effective team configurations. Using game-based approaches in non-gaming context has several novel advantages. First, it motivates people better than other known motivational methods. It has the ability to transform "*an intrinsic motivation with an extrinsic reward*" [156, pp. 27]. Secondly, we propose that a set of game components are used to build a game and its outcomes can be analyzed by using game theory.

4.3 Game Theory

Based on the idea of representing social situations in a game like setting, game theory is a useful technique for analyzing conditions, particularly where an individual's outcomes are depending not only his or her choices but also significantly affected by the decisions of other participants'.

Game theory was first introduced by Neumann *et al.* in 1944 with their famous book "*Theory of games and economic behavior*" [164] for economics, which was later adapted to several other fields such as sociology, biology, operational research, and computer science, etc. There are, however, the two branches of game theory, which differ in a number of respects, which are known as cooperative and competitive game theory. In the competitive form, participants of a game are considered as independent (competing) individuals, who aim to maximize their profits or interests regarding to a situation. On the other hand, the notion of cooperative game theory relies on the fact that there is a likely chance of quantifying the cooperation among the participants with *transferable utilities* (i.e. a commodity such as knowledge or money that can be transferable among participants), who aligned themselves with incentives for cooperation.

In knowledge-based activities, human teams are more capable of achieving collective success rather than relying on individual efforts (i.e. *knowledge is a socially constructed entity* [165]). Therefore, software development is usually considered as a collective work of *strategic* interactions [6]. Game theory is a mathematical theory of interactions, which allows us to model several social interactions such as the conflict of interests [166]. It focuses on the interactions of individuals and choices of people and their contribution to an outcome of a situation. Game theory is also useful for analyzing the past and the present situation for expected behaviors of participants as to their different individual payoffs or organizational outcomes [167]. In fact, game theory is a well-developed theory for describing the interactions of rational, independent agents in a variety of settings used for creating approaches in fields including, economics, computer science, social science, political science, and biology [168].

Game theory investigates the outcomes of interaction of entities. It is a collec-

tion of analytical methods or tools based on mathematical models to define or observe social situations (e.g. conflicts) [168]. To this end, game theory outlines interaction of people in terms of mathematical game forms. These forms consist of players (participants or actors), rules required for their interactions, actions or strategies (strategy profiles) of participants, and have definition of outcomes (i.e. payoffs) of their actions. One of the basic assumption in classical game theory is that participants are rational, i.e. follow the rules and play to win [166]. Common game forms involve these: (1) Non-cooperative games where participants only act according to their benefits, (2) Cooperative games where participants are inclined for cooperative behavior (i.e. cooperation is used as the main motivator), (3) Zero Sum games where one of the participants should win the game while other(s) lose, (4) Constant Sum games where the reward for each participant is constant [169].

4.4 Games in Software Engineering

The social and economic value of software development has become more prominent in an information based global economy. Therefore, as has happened in many other fields like sociology, economics or computer science, there are some approaches that are using the game theory in software engineering research.
Several limited attempts have been made to understand software development as a cooperative or a competitive game form. For example, Lagesse [170] built a model based on a cooperative game theory approach with the idea of optimizing task assignment in software engineering efforts. On the other hand, Grechanik and Perry [6] focused on a game theoretic approach as a non-cooperative game, based on the fact that there are a number of potential goal conflicts among the roles of a software development approach. Moreover, Cockburn [171] considered software development as a series of games of invention and communication, where he portrayed the software development as "*economic-cooperative gaming*". His vision is similar to an iterative game in which two goals are competing for a resource. He also suggested that as an emerging area, which he called "mechanics and economics of communication", it should be investigated in the near

future. As to the skills of the participants, Cockburn [172] also pointed out that software development should be considered as a game hinging upon its project resources. Using an approach based on grounded theory, Baskerville *et al.* [173] considered trade-offs and balancing decisions as balancing games that may appear in three different levels (i.e market, portfolio, management), where their nature is to progress dynamically with the demands of a market. Ko *et al.* [174] used a game theoretic approach to improve the reliability of data collected by using a method to improve its accuracy for a more effective quantitative process management, where they also recommended a study for applying game theory in software project management and software process improvement activities. To improve the learning abilities of students, Holeman [175] designed a software process improvement game, which is a type of board game (designed to instruct CMMI to students) wherein participants compete for achieving CMMI level 2 on a Monopoly-like game board. Ogland [176] developed an approach for conflicting situations by using game theory and drama theory. He portrayed SPI as a game, which is playable by quality auditors, software engineers, and managers. The goal is to identify how an SPI standard progresses through equilibrium (i.e. a proposed solution concept in a game).

Although game theory can be considered as a newly emerging field, there is a variety of related works highlighting the importance of decision-making in software development landscapes. Equipped with the idea of "making everyone a winner", Theory W [63] is an approach based on the concept of risk management in software engineering decisions. To resolve the conflicts among the stakeholders of a project, it also suggests that the role of management somehow acts like a mediator or a negotiator, which seems similar to a game theoretic approach. In order to establish *a value based approach* and formalize the design goals of software development, Sullivan *et al.* [18] considered software design as an investment activity, where they applied the concept of real options to evaluate economic outcomes. To improve the effectiveness software architecture decision-making, Vajja and Prabhakar [177] investigated design issues based on the quality attributes, where they can be modeled as a game

theoretical problem. Sazawal and Sudan [178] suggest a game model, *basic software evaluation game*, intended to be useful for helping software teams on decision-making particularly from an evolutionary perspective on software design decisions. Furthermore, they hypothesize that *lightweight* game theory is more useful for understanding software evolution. Bavota *et al.* [179] investigate the opportunities of using non-cooperative game theory of "extract class refactoring". One possible situation maybe when two players are competing to build new classes for improving the levels of cohesion.

Social dilemmas are situations that are arisen from the conflicts between collective and private interests (mostly long term versus short term). In other words, participants may discover a more feasible alternative action suiting their self interests better than team-based contributions. The prisoner's dilemma is a simple framework, which has been used frequently by researchers to observe conflicting situations. A set of studies in software engineering literature focus on the application of Prisoner's Dilemma, which is a non-cooperative game based on two persons interactions. For example, Hazzan and Dubinsky [180] investigate the way of cooperation in extreme programming, in particular for pair programming practices. Secondly, a hidden game of Prisoner‘s Dilemma is investigated by Feijs [181] between a programmer and a tester. Thirdly, Oza [182] uses Prisoner‘s Dilemma framework to investigate strategic interactions in a client-vendor relationship in offshore outsourcing projects. Recently, Klein *at al.* [183] draws out attention to the notion of incentive conflicts in software development both for identifying design characteristics and resource allocation perspectives.

From a software organizations perspective, these findings suggest that game theory is applicable to various software engineering problems. There is, however, a divergence between the theoretical and practical approaches. Therefore, the current challenge here is to find more practical ways to apply aspects of game theory in the software development process.

4.5 Mechanism Design

In this section, we highlight important points of mechanism design (MD). A subfield of game theory, mechanism design, specifically deals with social decisions and their effects on outcomes. In this framework, a designer or a manager investigates how one designs the social structure of an organization so that the individual incentives of participants can be transformed into the organizational wide desired goals. In other words, mechanism design is an approach to improving the social structure of an organization.

A mechanism functions as a communication model of an organization. Based on a set of rules, this model takes the required information from participants as inputs and determines a social outcome [184]. The notion of MD is about understanding the structure of an organization such as a communication system for improving social decision-making and societal welfare. In MD, a social planner can create an organizational structure to induce a planned or desired outcome based on the private information held by the participant's of an organization. The information provided in this process is useful for modeling organizational procedures, solving economic problems such as allocation of resources, or dealing with problems related with asymmetric information and ultimately for supporting cooperation among the organization [184]. MD should also assist a social planner in modeling an organization for analyzing how the private information of individuals interact throughout the organizational rules, which directly affect the expected outcomes. Such a model usually depends on what the possible action for each participant is and what consequences are.

Hayek [185] developed the idea of viewing social organizations as mechanisms for communication and information exchange. Hurwicz [184] introduces the concept of economic mechanisms to model organizations where participants communicate, and exchange information. Furthermore, he coined the term *incentive compatibility*, which ensures that self-interested individuals can be motivated to reveal their true preferences and their private information. He suggested that a model of an organization should be based on communications and actions that are available to each participant in an institution. Harsanyi [186]

developed a model based on the theory of Bayesian games, (i.e. games with incomplete information). He investigated situations where individuals have insufficient information. In particular, he studies cases where participants have uncertainty about other participants' information (while the rules of the game form is known by all the participants). Moreover, he worked on models of incomplete information based on issues of modeling participants' actions in terms of each others' in social organizations.

A number of research has been conducted on the application of MD to several information technology problems. For example, Zhao *et al.* [187] propose an approach to Internet security issues as economic factors such as factors governing the actions and interdependence of the participants. For this purpose, they implement an economic mechanism (in this context, a certification mechanism) to reduce the security risks of users over the Internet. The essence of this mechanism depends on the idea to minimize the possibility of sending out malevolent traffic by increasing the responsibility of service providers and promoting the incentives to monitor the suppliers of malware and spam in their networks. The mechanism best works on a certified network concept by which each certified service provider will be able to use the collected information from other providers and held responsible for the traffic that is generated by their users [187].

Stef-Praun and Rego [188] outline a simple mechanism to transfer system wide efficient allocations of resources rather than individual resource allocations in a decentralized market for web services producers and consumers. Authors claim that the proposed mechanism can be realized to fit any structure composed of a large number of self-interested participants (e.g. a dynamic collaborative environment). Friedman and Parkes [189] investigate a customer pricing problem of a wireless networking provider, which may be seen in a coffeehouse as a mechanism design problem. They develop a game theoretical model for bandwidth allocation based on a game of incomplete and asymmetric information

Taken together, these findings highlight that the MD theory and its actual implementation in software organization can be helpful for analyzing several social

and economic interactions. Furthermore, it may be used for building models for understanding software development team configuration. In general, a game based approach can be used for revealing true preferences of self-interested individuals by designing the rules of interaction e.g. communication protocols, game rules, etc.

4.6 A Game Theoretic Perspective

As previously mentioned, software development is inherently a complex process with a common element of uncertainty [190], which could primarily be caused by human factors. Conceptually, the development process is formed by the interaction among several participants who are performing a set of roles with distinctive personality traits. A number of previous studies actually have reported that interpersonal conflicts between these roles are unavoidable during the software development activities [191,192]. A reason for this is that the personality traits of software practitioners should have an impact on team based interactions.

From a social perspective, a software process can be considered as a workflow in information streams. Consequently, a software organization can be defined as combinatorial networks of people connected for software development teamwork. In order to address the adversities associated with people and their connections, it is important to investigate the potential conflicts among the management and the software practitioners who are traditionally known to be individualistic rationalists [6].

For example, while management aims to improve the effectiveness of software teams by using social characteristics of their personnel, a software practitioner may not be interested in revealing his personality type. This conflict can be investigated by using a game theory, which takes the strategic possibilities into account. The goal of a game theoretic model is to reveal the private information of the participants and combine their preferences to explore an optimal configuration.

To build a simple game to represent such conflict, both the manager and soft-

ware practitioner need to be identified by their private information, i.e. their possible strategies (the options he can choose from), and payoff possibilities for all participants.

4.6.1 A Conceptual Game Model

We consider a strategic game form:

$$G = \{N, (S_i), (u_i)\}, \tag{4.1}$$

with n participant i from a set of $N = \{1, .., i, ...n\}$ players. These players, moreover, choose a strategy s_i from a finite set of strategies S where a strategy profile is denoted as $s = s_1 \times s_2 \times s_3.... \times s_n$. Let s'_i be any action of player i, which can be either equal or different from s_i, therefore (s'_i, s_i) is an action profile where all players except i choose their strategy s. In addition to that, $-i$ signifies the participants other than the participant i (i.e. except i) whereas S_{-i} designates the strategies that are not chosen by participant i. The utility function for player i is denoted by $u_i : S \to \mathcal{R}$, which presents player i's payoffs. Such a game should be considered as *naturally strategic*. Normally, this means that participants' expectations of other players' decisions are likely to affect their actions. In light of these remarks, consider the strategy profile above is a solution to game $G(.)$ where a strategy s_i is a strictly dominant strategy for every player $-i$, which is assume to be a steady state among the players.

$$u_i(s_i, s_{-i}) > u_i(s'_i, s'_{-i}) \quad \forall s'_i \ \& \ s_{-i} \tag{4.2}$$

4.6.2 Two Person Game Form

Here, we describe a two-person game based on particular goals of a software practitioner (P) and a game master (GM). In the two person game model, we assume that GM has two strategies: (i) building the ideal team, which hypothetically consists of the participants who are socially compatible and therefore able to work with the best performance (ii) building the actual team, which is

composed of the individuals who are not gathered together by any social constraints such as their personality characteristics. Similarly, P has two strategies; (i) reveal his true personality type, (ii) not reveal his personality type.
The intersections of these four strategies define four different outcomes as shown in Figure 4.1.

		Game Master (GM)	
		Building the ideal team	Building the actual team
Software Practitioner (P)	Reveal himself	P reveals, GM builds the ideal team 2,4	P reveals, GM builds the actual team 1,2
	Doesn't Reveal himself	P does not reveal, GM builds the ideal team 4,-1	P does not reveal, GM builds the actual team 3,1

Table 4.1: *Outcome Matrix of the Game:*
Key: (x,y) = (P,GM) → 4 = best; 3 = next best; 2 = next worst; 1 = worst; -1 = improbable

In our two person game form; N = {P, GM }, S_1 = { Reveal, Not reveal }, S_2 = {Build the ideal team, Build the actual team }.
The game is shown in matrix form where the outcome matrix is represented by an ordered pair of numbers, x representing P, and y showing the preferences of GM. We assume that the value of outcomes represents ordinal preferences (i.e. 4 is the best, and 1 is the worst).
The ranking values found in the outcome matrix are based on the assumption of the goals of the two players.
Game Master (GM):

- Primary aim: Building the ideal team.
- Secondary aim: Revealing software practitioners personality type.

Software Practitioner (P)

- Primary aim: Does not reveal his true personality type.
- Secondary aim: Wants to work in an ideal team.

The primary goal of a GM is to build the ideal team using the personality type information. However, to this end, GM needs to reveal the personality types of practitioners. Although we propose several rules of interaction to regulate the game setting and to assist the GM in the process of identifying the personality characteristic of a practitioner, we know that P may not prefer to reveal his true type. Note that if P does not reveal his personality type, it would be impossible for GM to build the ideal team (see (4,-1)). In light of this, a stable outcome for this game would be (3,1). However, we seek to build an ideal team, thus the rational (desired) outcome for the game for us, which should be (2,4).

4.7 Game Composition as a MD Problem

Consider a software development project where a set of individuals should be organized to improve the effectiveness of software teams as regards their personality types. A manager (social planner) facilitates the activities of team composition where the participants of a software development project are assigned to their ideal teams. Next, based on the participants' personality types, a manager needs to decide whom should be appointed to which team. This situation can be understood as a mechanism design problem where the goal is to understand effective team structures so as to maximize the team efficiency. A conceptual solution to the problem of understanding efficient team compositions in a software development organization can be solved by

- Identifying the participants to true personality types;
- Exploring the effective team structures by using this information.

4.8 Rules of the Game

We formalize the team configuration mechanism with the assumption that we can identify the personality traits of individuals using a psychometric assessment approach such as MBTI. Hence, we design the rules of our game. Traditionally, a game should be managed by an individual, the game master (GM), who can be a software manager or, team leader. It could be played by either a single person or a software team. In this context, the expected outcome of the game is the true personality types of individuals where the primary job of a GM is to orchestrate the game, for example, by asking questions to the participants. The rules of the game can be itemized as follows:

- GM announces the participants how the game is operated and shows one special type of form, which is later used to compute the personality traits of an individual.
- GM sets the length of the session, i.e. thirty minutes ideal but may change with respect to size of the software team.
- After giving the instructions to the participants, GM draws a card from a card deck preferably within a sequential order (i.e. a game deck shall set up with a specific rule).
- GM shows the picture of the card and further reads the situation on the card with two different answer choices. Participants, then, are asked to select between two possible responses.
- GM waits for all the participants to finish marking their answers and continues with the next question until the full deck is done.
- The game mechanism reveals the personality type of the participants (for each individual based on the acquired knowledge) regarding the patterns of their behaviors.
- GM informs the management about personality characteristics of the players, i.e. software practitioners.

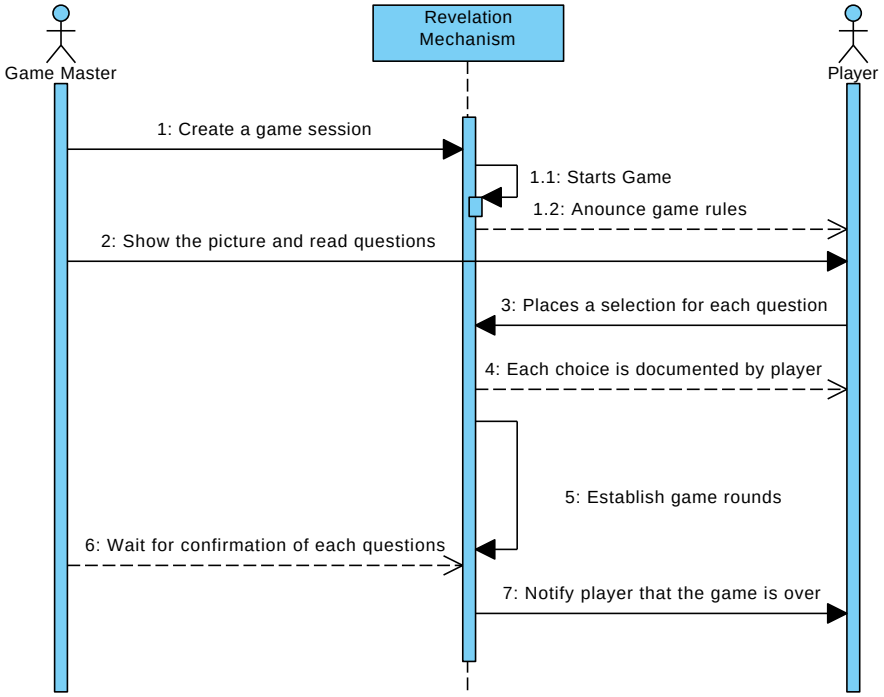


Figure 4.1: *The Sequential Steps of the Game of Revealing Personality Traits*

Figure 4.1 shows the graphical representation of the game of revealing personality types. It is apparent from the figure that there are two main participants: the game master, and a player (or a team of players). The revelation mechanism is the formalized rules, which are designed for mediating the interactions between the game master and the player(s). The mechanism can store the data in a paper form or otherwise the game could be realized as a software system where the game master and the mechanism could be implemented using a software.

4.9 Chapter Summary

Software development required teams of self-interested individual actors to contribute effectively to organizational goals. Organizations that have failed to account for the motivations of individual participants often experience difficulty accomplishing their goals [193].

The concept of gamification has been successfully used in different settings including but not limited to organizing training programs, maintaining personal activities, online and in-person shopping, conducting market research, etc. [156]. We suggest that the gamification process is applicable to any organizational setting that develops information or knowledge artifacts (e.g. documents, source code, etc.). In particular, it is likely to be more effective in volatile environments that need social interactions among its participants where the rules of interactions dramatically affect the process of artifact creation and ultimately organizational productivity as a whole.

Game theory suggested that we viewed the organization and its goals from the standpoint of individual rational actors, who did choose actions that maximized their expected utilities, subject to their incomplete knowledge of the motivations and likely actions of others, and their limited ability to predict future outcomes. The challenge for software organizations was to allocate resources and to create incentives and disincentives in such a way that participants were motivated to take actions that contribute effectively to organizational goals.

In the next chapter, we introduce the notion of software ecosystem, and continue with social and value dynamics of software development. Later in the next chapter, we will detail the software development productivity, the software artifacts, and social productivity concepts.

Chapter 5

Social and Value Dynamics of Software Development

5.1 Introduction

This chapter intends to set a background to social and value dynamics of software development. First, it reviews the notion of software ecosystem, software artifact and the value perspectives. After reviewing the literature of software development productivity for identifying its factors, a number of software development productivity models are developed. Next, a group of social capital models are built based on a model of social capital (i.e. [194]) and illustrated accordingly. In the final part of this chapter, the concept of social productivity is introduced, and lastly a set of models for the social productivity of software development are constructed based on the factors extracted from the software engineering literature.

5.2 The Software Ecosystem

In recent years, exploration of the importance of the interactions of an economic community has highlighted the fact that software development organizations should co-evolve their capabilities and roles for maximizing the opportunities for project and business success. Therefore, the traditional viewpoint of a

software business - selling software to the mass market - has been replaced by the idea of organizations, which are formed by interacting entities similar to a biological ecosystem [195]. Based on the idea that interacting participants and organizations of the business world is considered similar to mechanisms of the nature,

Moore [196] introduces the definition of a software ecosystem as an *economic coating* that forms around a software product. Parallel to Moore's definition, Mitleton-Kelly from the London School of Economics investigates organizational complexity by applying the theory of complex social systems to the theory of organizations [197]. She suggests that complexity arises from the interactions through the elements of a complex co-evolving social ecosystem, including all individuals and organizations based on their business, technical and organizational relations among suppliers, customers or competitors.

According to Shapiro and Varian, "*there is a central difference between the old and new economies: the old industrial economy was driven by economies of scale; the new information economy is driven by the economics of networks*" [198, pp. 173]. Therefore, a software ecosystem should be based on various information exchange networks. It must be considered as a set of several business entities working on collective outcomes in a shared market where several entities play distinctive roles. The relationship is based on the exchange of knowledge in terms of several forms such as information artifacts. Recognition of the software development organization as a social ecosystem brought the realization that the investigation of its social structure (e.g. connectivity, cohesion or coupling of its members) may help to improve the human centric aspects of the business process.

5.3 Social and Value Dynamics

Social dynamics, also known as the dynamics of human interactions, is a multidisciplinary field of science that is concerned with analyzing socialites or social systems formed by participants and their interactions. This section surveys several important concepts and definitions and the foundations of social and

the value dynamics of a software organization. These concepts and definitions highlight the important points of the Social Aspects of Software Engineering (SASE) [5]. Ultimately, SASE will help us to understand social and value dynamics of a software organization, to promote cooperation within software teams and organizations and thus to make them respond better to the dynamic and future trends of software development. We start this section by defining a software artifact. Next, we identify sources of capital that are used in any production process. Moreover, we define both social and human capital, as well as introducing the concept of social productivity.

5.4 The Software Artifact

The cost of quality attributes in the software development activities is heavily based on interaction skills of individuals and teams. Specifically, one of the most important of the output of these skills is the *software artifacts*. Software artifacts are defined as; "*The products, process and software developed by human efforts...that embody human knowledge.*" [199, p. 11]. Some researchers suggest that a software artifact has a social dimension because it is an outcome of a social process [28].

According to Baldwin and Clark, an artifact is a *quintessential* outcome of both human intelligence and endeavor. Nevertheless, knowledge-based artifacts (e.g. software, computers, etc.) are interconnected group of entities usually created by a team of workers [200]. Morisio *et al.* point out that the artifacts produced in a software process are complex creations and channeling of the human acumen identified several different characteristics.

Tsui [28] describes the notion of software artifacts as the a "unit of material", which can be in any form such as documentation or source code; its life-cycle starts from requirements analysis phase and follows through product development and documentation. Several entities can be accepted as software artifacts including manuals or guide books, and even internal deliverables inside the organization. Software artifacts are considered as smaller and manageable parts of a software project. They are useful *touchstones* for implementing the con-

cept of *separation of concerns* [201], which values the division of the effort and knowledge by coordinating the software engineering tasks and decisions [200]. Shariq suggested that the knowledge should be considered as an outcome of human activities, which essentially produces *knowledge artifacts*, and knowledge networks are intervened by these artifacts [199]. Cluts conducted a case study to develop a framework based on the connections between people and their activities where artifacts are described to contain a backlog of the past events and connections among them [202].

5.5 Productivity

In so far as it is not different from other forms of production, software production is considered as an economic process of conversion of inputs to outputs based on industrial methods of production. Consequently, one of the concerns of industrial process improvement is investigating methods to improve and measure the economic productivity. However, the social and economic aspect of productivity should depend on a set of several related factors, which will be explained in the following subsections.

5.5.1 Economic Productivity

In general, *economic productivity* is considered as a value to measure the efficiency of this production process. For example, it is measured as a ratio of the units of inputs versus the units of the outputs [203].
However, it is also considered as a utilization of resources with an optimal cost [204](a ratio of production capacity to production cost). It depends on the availability of resources and is highly connected with the *value* creation processes [205]. Brynjolfsson [206] states that productivity is an essential economic criterion for the contribution of any technology to an economy. Sink *et al.* [207] defines productivity as a ratio between the *actual output* versus the *expected resources* that have been used. Based on the assumption that time is a resource, Jackson and Petersson [208] suggest a time-based measurement of

productivity (i.e. a ratio between value adding time versus the total time). The limitation of this approach is that usually there could be a lack of information about the resources that are consumed during the production process.

5.5.2 Software Productivity

Similar to several other industrial propositions such as industrial and manufacturing processes, software productivity is traditionally defined as a ratio between the inputs (e.g. the cost of work/resources) versus the outputs (i.e. software artifacts or services) within the production process of software development [17, pp. 153]. This ratio could be considered as an economic output (e.g. lines of code, function points, etc.) divided by the economic input (e.g. resource requirements, personnel skills, etc.), which will eventually contribute to the completion of the end product [209]. Traditionally, a number of researchers use the size of the software as a primary measure for software development productivity [210]. Several software size measurement methods exist where the most common metrics are lines of code, where it was suggested as a common measure for size and complexity of a software [211], function points [212], which favors the user perspective for assessing the functionality of a product, function points per hour [213], and measurement of effort [214].

However, empirical evidence suggests that it is hard to find a suitable way for measuring productivity [215] in industrial production and software development productivity in particular [10]. This view is also supported by Zelkowitz [210, pp. 7], who states "*As software development productivity is a function of software size, this makes comparisons of software productivity across organizations and countries very difficult*".

In fact, software productivity is considered differently for stakeholders from their distinctive perspectives. For example, from developers' viewpoint, a productivity measure could be the amount of code produced for the software system; on the other hand, from the users' perspective, it could be the degree of functionality achieved for the software system.

The broad use of the term productivity is sometimes measured from different

viewpoints such as the skill set of software practitioners, their techniques and the instruments they used in the software development processes [216]. From an industrial point of view, productivity is generally understood as a key issue for software development organizations when creating a competitive advantage. For example, it is vitally important to reduce time to market of a product while concurrently maintaining the quality of the product. Trendowicz and Munch [217] suggest that the factors affecting productivity of software development should be selected based on the economic significance of their attributes, which could also alternate in different domains of software development. In addition, they claimed that a productivity model should only include the factors, which are found as the most important ones by the literature.

According to Jones [218], software development teams that are building similar kind of artifacts can easily progress to more mature stages on software productivity because it improves the experience levels of both managers and software teams. Furthermore, he argues that reuse could have not only a positive impact but also a negative impact both for deliverables as well as productivity improvement efforts as a whole. For example, using high quality reusable artifacts could improve productivity; however, this can also reduce the productivity of a software project because such an artifact may not be near zero defect levels. Productivity is significantly affected by the quality of workforce, management capabilities and environmental conditions of a software organization [17]. Moreover, the effective usage of methods and processes, project complexity, software team morales, and effective team configurations are the key adjustment factors for software development productivity [218]. However, interdependent factors involve with productivity cannot easily be controlled or improved by only manipulating the variables such as dynamic motivational factors, cost of communication and social expenses [219].

5.5.3 Software Productivity Improvement

Several software engineering researchers suggest methods to improve software productivity by balancing the field of tension among people (regarding to their activities), processes (with respect to its tasks) or technology (by its advances in computing power) [220–222]. There are several attempts in the literature at understanding and measuring software productivity. For example, Scacchi suggests a framework for examining and measuring software productivity to perform a simulation over the *production dynamics* of software projects [221].
One common approach to improving software productivity relies on the theory of group productivity by psychologist Ivan Steiner [223] who states that consequences of defective processes are important for explaining actual productivity [224]. It is calculated by subtracting these defects from potential productivity (i.e. Actual Productivity = Potential Productivity - Losses Due to a Faulty Process). Abdel-Hamid [224] explains potential productivity as follows; if an individual or a group uses the maximum potential of its resources, then a level of maximum productivity is achieved. He adds that two factors are important for representing the shortfalls for software quality and productivity problems; (i) the task's characteristics (i.e. complex nature of a task) and (ii) team resources (i.e. fitting individuals or team skills over tasks and tools). These factors could increase the cost of communication and lower the motivation of individuals and software teams.
A common view in engineering terms is that the productivity improvement indicates producing more outputs from a known set of inputs by reducing the influence of any factor that hinders productivity. For example, software productivity improvements can be achieved by having a skillful team, improving the path of development by reducing rework, and by creating reusable and more manageable software artifacts [225]. In fact, an increase in the productivity is achieved when human resources used in the software development process starts adding more value to the software product.
Over the past few decades, software productivity has been investigated by using several indicators affecting the productivity. One such approach is conducted

by Pfleeger [226] who uses a statistical method called regression analysis. By using this technique, he constructs an estimation model of productivity where he calculates the effects of cost factors in a predictive manner. Moreover, regression analysis has also been applied for determining the correlation between size and effort for software development projects [227].

5.5.4 Factors of Productivity

Although a considerable amount of literature that has been published on productivity factors affect software organization [228], several questions remain unanswered (e.g. a correlation and/or significance among these factors). For example, it is considered hard to address a single solution to the identified issues of software development productivity [225].

Team size is another important factor for understanding the increase in development productivity as well as project size and complexity. However, researchers have not dealt with team size in much detail. Moreover, creative and talented individuals are the main assets of productive teams in software development projects [229]. This suggests that neither of the other factors can produce more significant weakness than a lack in human capital.

One of the first systematic studies of the software development productivity issues was reported by Scacchi [230], who reviewed the entire software development productivity literature while analyzing potential productivity problems. First of all, he suggested that a multivariate analysis for identifying the factors affecting the software development process might be essential. By way of illustration, he reported several measurement issues; (i) measuring the lines of code may not significantly reveal the true value of productivity, (ii) changing productivity patterns should be observed and factors for productivity improvement should be revealed, (iii) the consequences of multiple changes from different stakeholder perspectives like managerial versus technical should be understood [230]. In addition, he also mentioned that in middle or large scale projects, average skilled personnel might not be important for productivity, while in small projects skillful individuals might have more significant effects

on the overall productivity. One of the most significant points he made is that the software development productivity is directly affected by the kind of methodology selected (e.g. iterative or incremental).

A large and growing body of literature has investigated the factors affecting the productivity of software development organizations. By following the software development productivity literature that was summarized by a systematic review [228], we select the factors that potentially affect the productivity of software development. These are (i) the software development process [10, 221, 225, 231], (ii) the level of individual's motivation [10, 16, 225] and its influence on software engineers [232–234], (iii) the ability of an organization to stabilize the customer requirements [213, 235], (iv) software project management quality [4, 236], (v) team issues such as aligning skills of the software teams [221, 225, 231], (vi) reuse [16, 225, 237–239], (vii) tools that are used in software development [240, 241], (viii) the effect of programming language on software development productivity [16, 225, 242], (ix) software size [243, 244], (x) team size [240, 241, 243], and finally (xi) software complexity [10, 16, 225, 231].

Based on the factors reviewed from a systematic literature review [228], we illustrate Figure 5.1, which shows the conceptual model for the factors affecting the productivity of software development organizations. Detailed information on the development of the model, and a sizable majority of the studies surveyed here can be found in [245].

5.6 The Economic Value of a Software Development Process

A software development process aims to create an economic value for all the investors in the enterprise. Boehm [22] claims that many software engineering projects are considered to be performed within a value neutral setting. In other words, every task and activities are regarded equally important without considering the outcomes and business value propositions. However, researchers suggest that many reasons (e.g. lack of utilizing project resources, fail to prioritize project requirements) that cause software projects failures could stem from the problems of value-neutral approaches [14].

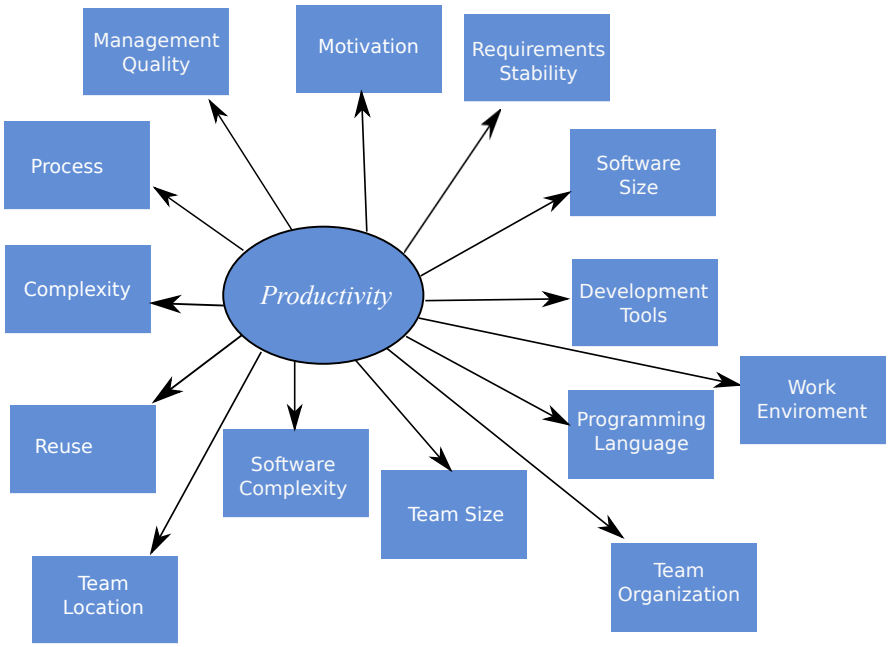


Figure 5.1: *A productivity Model Based on Factors Affecting Software Development.*

According to the software engineering perspective, value creation activities mostly focus on the economic significances; e.g. customers' requirements and the things stakeholders are valuing the most [17]. However, the stakeholders who contribute in the value creation process have different considerations and therefore different goals and expectation are required from the same software system. Furthermore, they might have subjective definitions of the value. Halling *et al.* considers the relationship between value and the project attainment and defines the goal of a project as producing greater value than the values of the resources consumed by the investment in the software organization [23]. Boehm and Sullivan suggested that the best way to establish the uttermost value from software project resources is by administering the software development process as an *economic activity* of investment [246]. The knowledge which is accumulated as the capacity of an economic activity is ultimately based on the human capital, which directly influences the efficient use of physical capital [247].

5.7 Human and Social Capital

The classical notion of capital states that the capital becomes apparent from the social interactions between capitalists and laborers. In other words, it is an end product of a *social process*. According to Marx, it is a *surplus value* captured by individuals who control the production processes [248]. In addition, it is also a kind of activity of investment for the resources so as to gain profit. Therefore, capital is not only a result of the process of manufacturing but also an outcome of trading products and goods based on the social relations between capitalists and laborers [248].

In the last decade, this classical viewpoint has evolved to include the intangible assets for human intensive organizations such as the economic value generated by human and social capital. Understanding and measuring human capital is a challenging process, evidence suggests that quality of social and organizational relations based on several individuals' interaction affects the sustainability of any social structure. Human capital theory relies on the fact that laborers become capitalists by accumulation of knowledge and skills and therefore, human experiences are embedded inside the notion of capital [247]. It simply states that the laborers who are trained in specific subjects and captured valuable experiences in their work life somehow become irreplaceable through the production processes, which also constitutes competitive advantage for a software development organization. One form of human capital encompasses several intangible assets such as the personal social network of resources of an individual is also known as social capital.

Social Capital can be defined as an intangible resource, which benefits from social connections and networking. It may include the opportunities that an employee's social network can provide. Lin defines social capital as "*investment of social relations with expected returns in the market place*" [249, pp. 19].

Bourdieu understands the term as a presentation of actual and future resources that are linked as a network of relationships [250]. His definition designates that social capital is based on two components; *social relationships* which afford possibilities to help them obtain accessibility to the resources by their

relationships, and *resource quality*. He claims that the value of social capital, which is based on social connections, should easily be convertible to an economic form of capital.

There are other definitions of social capital [251]. Some are (i) a resource that individuals yield from social structures regarding to quality of their relationships, (ii) an observable pattern in a social structure which influences the relationships among the individuals or social groups, (iii) the quality of personal contacts which individuals gain to increase both financial and intellectual capabilities [249–251]. Fukuyama defines the social capital as "*the ability of people to work together for common purposes in groups and organizations*" [252, p. 10]. Later, in his works, he considered the term as *an intangible value obtained from social groups* that promotes collective outcomes. He argues that social capital is dependent on norms like honesty, trust and dependability.

In the field of social sciences, a social network is an organized form of people that comprises the individuals and the connections among them. In general, individuals are considered to be connected in a fabric of social network, and expect some benefits from the social formations and the way the network operates [253]. Consequently, social capital may be broadly defined as an intangible resource accumulated by the social connections. Therefore, individuals should have to be linked to one other to improve their social capital.

Social capital comprises resources to be captured by individuals [254]. According to Portes, social capital is inherent in the fabric of actors and relationships. In order to own a social capital one should be linked with others, therefore, it should be measured with quantity of social connections that an individual might have [251]. Coleman concludes that all kinds of social structures, henceforth relations, enable some form of social capital. In fact, the individuals intentionally connect with one and other to form social networks and expect benefits from these actions [255].

In this thesis, we define the term as;

> ***Social capital is a multi-dimensional latent construct, which usually found in the potential form of intangible resources based on patterns of social connections and social skills of individuals, teams or social groups who has the ability to contribute to the economic progress of an organization.***

The higher level of social capital attainable by participants of a software development organization should help to improve the productiveness of teams and individuals in a software firm. Consequently, leveraging the social connectivity in a software development organization shall have positive impact on the productivity in a software development group. Aligned with the improvement efforts, this can be considered as one of the actual benefits of social capital obtained from networks of relations. Exploring and implementing team based social improvements will help us to improve structural and organizational stability. It therefore enables us to constitute more cohesive information exchange networks, which may have a positive effect on the productivity of a software team.

Based on its qualitative attributes, social capital is a network of elements consisting of nodes and links of connection. Hence, this form of capital can be improved by creating productive patterns of social interactions. At a social level, it is not surprising to discover that *social capital* can be transformed to measure the productivity of a team or an individual of a software development organization. In light of this information, it should be easier to create compatible and productive team formations. The value of social capital is mostly hidden in a network of interactions or connections. Hence, it could be observable in the social activities of a software development organization.

Coleman [255] suggests that all kinds of social configurations may create some amount of social capital. However, to gain a benefit from their existing social capital, its relationship with social productivity should be investigated.

As previously described in the work of Narayan and Cassidy [194], we build

a social capital model by using the illustration of seven dimensions of social capital as shown in Figure 5.2.

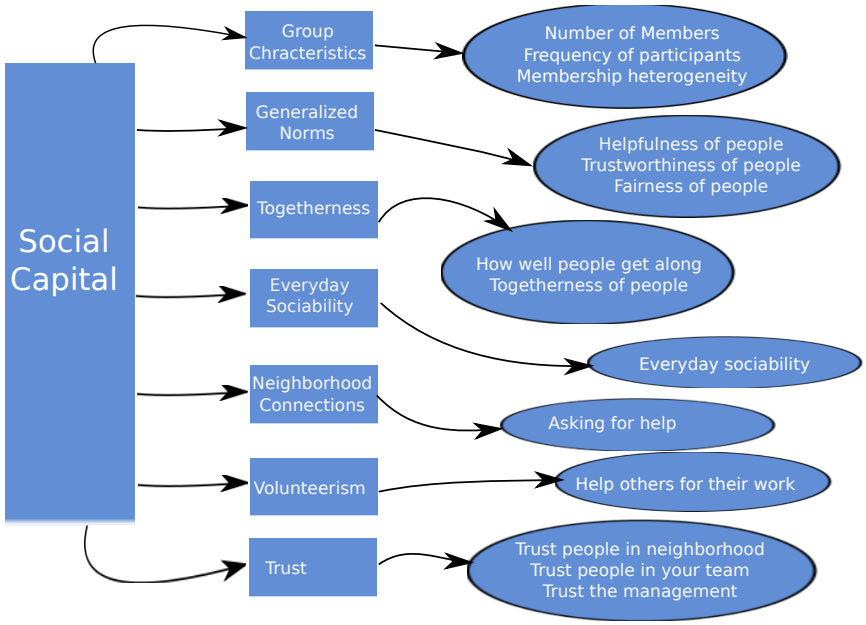


Figure 5.2: *A Model of Social Capital*

5.8 Social Productivity

Humans are *social* creatures. This means they usually depend on others and prefer to live in interacting groups (or socialites) where they influence one another. In fact, they continue to be increasingly interested in establishing a society and improving social outputs of their organized groups. Thenceforth, they prefer to work in teams and are inclined in order to form more complex outputs. By considering the social behavior itself as a method to exchange goods [256], they create and share knowledge-based outcomes (in forms of artifacts), and have their experience pass through further generations so as to improve the economic well-being of a society. Bourdieu argues that a sociality is circumscribed both by the available information and socially structural establishment of human mind [257]. It is therefore not surprising to discover that there is a strong correlation between a concrete social structure and a productive group from a socio-economic perspective.

For many researchers, productivity comprises the economic concept, however, it also has a sociological aspect which is highlighted by Barnett "*While an economic concept of productivity is undeniably important in explaining the material wealth of groups, personal observation suggests that understanding organized groups-including business firms-requires a sociological concept of productivity.*" [258, p. 739]

Moreover, Barnett claims that social productivity occurs when a team or group of people interact and create social interactions and outcomes, which certainly affects the functioning of teams [258]. It should also portray the actions and reactions of a social organization. In addition to that, he also describes *social productivity* as an outcome, which can be provided from a social group activity. As previously mentioned, software development is considered as a social activity where people should be working in close proximity. Therefore, the notion of social productivity should measure the level of this interaction. The economic perspective suggests that individuals' actions are established, directed and limited by interpersonal trust, social networks and organizations [255]. These interactions are based on group needs, values and actions of the group and further shape new actions or action sets. Consequently, Barnett indicates that there are mainly four constructs of *social things*, or main social outputs (matters and varies among groups) an organized group produces [258]: (i) *reputation*, in which a team is judged by its reputation which shapes its treatment by others, (ii) *symbols* that can be used by organized groups also designate symbolic functions for representing ties or ideas among groups, (iii) *trust*, which is an expectation of an individual of others (teams or individuals) to work for the benefit of the team as a whole, and (iv) *perceptions of fairness*, which state that people receive benefits with respect to their proportion of effort [258].

Therefore, social productivity is a vital component for understanding the structural complexity in a society. All constructs defined here are, however, can be considered as *resources of a group* ("ingredients of social capital"), as well as the outcome of functioning of a group ("features of social productivity") [258]. To understand the impacts of social issues over a software organization, we

investigate the level of importance for several social factors such as trust, communication, social life, and information awareness. We argue that social productivity should be materialized by several social factors where its relationship with the social capital should also be investigated. Here, we define the social productivity of software development as follows;

> ***Social productivity of software development is an intangible asset as we termed here to reify the effects of social factors on the social and economic landscape of a software organization. Therefore, a new kind of productivity improvement should be considered as the transformation of social capital (potential energy) into social productivity (kinetic energy) form.***

In other words, social productivity represents an identified stock of social capital that is transformable to value creation activities so as to form software artifacts. The notion of social productivity seems useful for achieving software productivity improvement goals. From a socio-economic viewpoint, it investigates ways to improve collective outputs, which enable a software development organization to make economic progress. These organizations build on the idea of collaborative social activities, which could be an identifiable component of teams that work in the favor of software organization.

In the socio-economic landscape of software organizations, social productivity should represent a concept for advancing the ability of software development organizations by understanding the factors that hinder social development and structure. It is, therefore, important to seek ways for increasing the efficiency and productivity of individuals, which depends on the subset of various factors mentioned below such as quality of their social interactions, and communication effectiveness of its members for their contributions to collective outputs, etc.

From a software development organizations perspective, social productivity is an attempt to explain the social factors that are hindering the software development productivity. Therefore, we select several potential factors affecting the

social productivity from the literature and build our hypothetical model (see Figure 5.3) based on these; (i) Stober and Hansmann [259] for reputation of a team leader on conflicts and his or her skills, (ii) Dittrich [5] for identification of communication issues with respect to level of individuals interactions, (iii) Koh and Maguire [260] for awareness of information in turbulent business landscapes, (iv) Hazzan and Dubinsky [261] and Anderson [262] for identifying trust in the software teams, (v) Kelly [263] for socialization or social life in the work environments, (vi) Hazzan and Dubinsky [261] for fairness, e.g. fair allocation of work, and finally Churchville [264] for frequent meetings i.e. how team members are informed about each others progress.

Figure 5.3 illustrates the model we propose based on the factors affecting social productivity of software development.

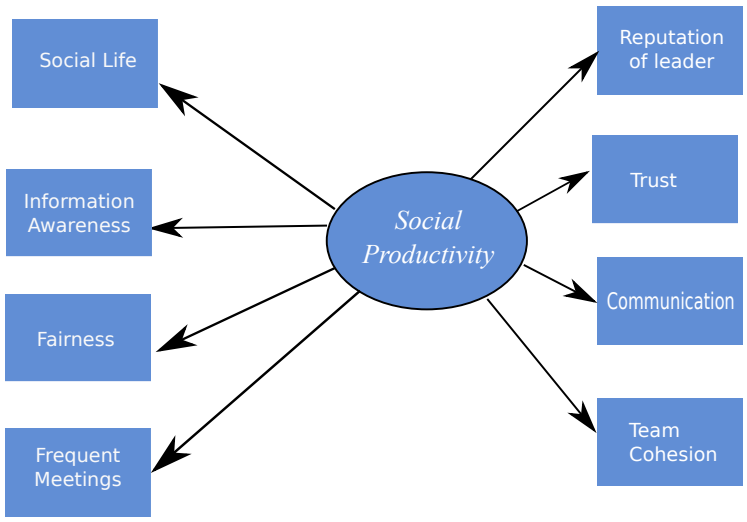


Figure 5.3: *A Social Productivity Model Based on Factors Affecting Software Development.*

5.9 Chapter Summary

In summary, software productivity is heavily dependent on the social aspects of productivity which can be achieved by better *social alignment*, i.e. matching the roles of people better to their personality types for maximizing their productivity. In addition, it is the skills of individuals and teams that transform the acquired knowledge into software artifacts (e.g. source code, documentation, etc.) and constantly increase the competitive advantage.

In this chapter, we introduced several definitions concerning social and value dynamics of a software development organization, and surveyed the literature to highlight several factors affecting productivity of software development organizations with respect to the value dynamics and several forms of capital namely social capital and social productivity. Based on the literature, we built conceptual models of productivity, social productivity and social capital. The goal was to identify several factors affecting these social constructs. In the following chapters of this study, we will measure the relationships between factors and the social constructs.

Chapter 6

Roles and Personality Traits

6.1 Introduction

This chapter surveys the roles in selected software development methodologies, which is followed by an attempt to build a comparison chart for these roles. Further Jungian personality traits are introduced and a review of the literature on the personality traits research particularly conducted in software engineering domain is documented.

6.2 Roles in Software Development Processes

Many different variants of development models and methodologies have been created. In this section, we survey the roles that are defined in the literature starting from traditional software development and working through ISO/IEC 12207, and agile methodologies such as extreme programming (XP), scrum and feature driven development (FDD). The selection of these methodologies were based on the following constraints: (i) the industrial popularity of those models in the software development landscapes, (ii) the academic popularity that are mostly mentioned in the academic papers. Ultimately, these development models become business and de facto standards for software development organizations.

6.2.1 Content Analysis of Software Development Roles

In this subsection, first we survey the literature for the roles for both traditional and agile methodologies that are mentioned in the previous section. We conduct a thematic content analysis (i.e. descriptive presentation of this literature review) based on roles as the units of analysis.

Content analysis is an organized study of characteristics found in a content of any type of communication, such as books, websites, newspapers [265]. Our approach uses the content analysis technique for making interpretations to create a role selection schema based on literature of roles in software development methodologies. Based on the survey data collected previously, these roles will be systematically compared to their industrial actualizations.

To this end, we first collect data from literature and rigorously classify them. We form a number of acronyms based on the roles that are found from the literature. Here, we are making partial use of a coding mechanism to construct a role-based schema with the defined roles from the literature. The coding aims to create variables based on the roles defined in software development. It is done for easy comparison of roles by constructing a unique key for each role found from the literature. Our coding schema allows us to observe the commonalities and differences between software engineering roles. It helps us to investigate the cause-effect relationships, interrelationships, and situational conditions for each role category. Here, we design several questions to seek validity for our coding in the defined categories, and analysis of identified roles from the literature.

- Is this role the same as a role in the other categories?
- Are there any duplicated role codings in a category?
- In which context do these roles emerge?
- What kind of roles have changed or evolved in the emerging methods?
- Is there any observable change for other roles when a role evolved to an other form (i.e. covariance between categories)?

The *objective coding* [266] is a technique to review a collection of documents for extracting and indexing the information so as to form a new perspective on representing the data. We use an objective coding scheme on the collected information of roles. This coding is helpful for visually comparing the actualized roles systematically with the ones cited in the literature. In the following subsections, several tables with assigned codes are built and ultimately a diagram is drawn to explore the relationship among roles.

6.2.2 Roles in traditional software development

Software engineering teams address the complex problems of software development by sharing the tasks among its members with respect to their roles. Roles are the descriptions of duties or assignments and competence for participants that are required to achieve defined tasks and activities of software development [43]. In his essay, *The Cathedral and the Bazaar*, Raymond states that because of the strict roles defined in the traditional software development, traditional approach is similar to building a cathedral, where a small team of people work in an isolated environment [267]. Therefore, this could be considered as a drawback because several artifacts are only visible to a limited number of individuals in this setting.

Code	**Role Name**	**Primary Type of Value**
PPM	Project Manager	Resource Allocation and Budgeting
SD	Software Developer	Development Activities
ST	Software Tester	Creating Test Plans
UID	User Interface Designer	Design Screen Interfaces
DD	Database Designers	Data Modeling
SAR	Software Architects	Software Modeling
BA	Business Analyst	Stakeholder Management
RE	Requirement Engineer	Gathering Requirements
SQA	Software Quality Assurance	Creating and Maintaining Quality
SAN	System Analyst	Construction of a System

Table 6.1: *Traditional Software Development Roles*

Traditional roles includes the following: *Project manager* who is responsible for allocation of resources, project expenditures, and responsible from the general objectives of a software project. A *software developer* is responsible for designing and maintaining the software programs, whereas a *software tester* is responsible for creating test plans and testing the developed programs. In many

cases *user interface designers* (design screen interfaces), *database designers* (design database schema) and *software architects* (design technical blueprints) are also included as a generic software practitioner category. A business analyst is not only responsible for solving the problems by regulating the connections between the business and the technical people but also for documenting several parts (e.g. requirement documents) of a software project. In addition to these roles, some others can also be seen regarding several needs; e.g. requirements engineer, systems analyst, software quality assurance engineer (see Table 6.1) .

Code	**Role Name**	**Primary Type of Value**
RO	Requirements Owner	Understanding Need
SDR	System Designer	Accomplishing work
SA	System Analysis	Reducing Risks
VV	Validation & Verification	Mitigating Risks
LO	Logistics and Operations	Understanding need
G	Glue among the subsystems	Accomplishing work, Reducing Risks
CI	Customer Interface	Understanding the Need
TM	Technical Manager	Technical Management
IM	Information Manager	Knowledge Management
PE	Process Engineer	Managing and Understanding Needs
COR	Coordinator	Organizational Management
CA	Classified Ads SE	Accomplishing Work (assumed)

Table 6.2: *Systems Engineering Roles and their values from [268]*

Sheard [269] identifies twelve roles (see Table 6.2) of development from the system engineering viewpoint while investigating the relationship between the roles and their importance for creating a value. This work not only suggests that the value is asserted in qualitative terms and it should be quantified in further research but it also claims that it should be observed as a requested improvement within a product by better (i) definition of the requirements, (ii) management strategies, (iii) ways for mitigating risks, (see [268] for details).

6.2.3 Roles in ISO/IEC 12207

ISO/IEC 12207 [46] has three main groups of roles for its participants. The first group consisting the principal roles are the *acquirer*, who is a form of stakeholder that obtains products or services from *supplier*, who is an individual or another organization agree on providing a software products or services. The *Implementer* executes development tasks, while the *maintainer* can be either an organization or an individual who performs the upkeep of developed

software; and *operator* is responsible for the execution of a system [46]. The second category consists of configuration and supporting roles: the *configurator* is responsible for the establishment and transformation of the information needed by an individual or a group; the *evaluator* tests and measure a software process or a product by using the data collected during the actual tasks that are performed; the *auditor* investigates the products and processes' compatibility with the agreements; the *usability specialist* deals with the demands and needs of the stakeholders such as the design activities based on human factors and skills and their fulfillment [46].

Code	**Role Name**	**Primary Type of Value**
AC	Acquirer	Software Client or User or Product Owner
SU	Supplier	Software Producer, Product Seller
IMP	Implementer	Realization of Development Tasks
MN	Maintainer	Maintain the Software
OP	Operator	System Execution
CFG	Configurator	Accomplishing Work, Reducing Risks
EV	Evaluator	Test & Measure a Process or a Product
AU	Auditor	Contract Management
US	Usability Specialist	Problems Regarding to People Factors
MA	Manager	Managing
AM	Asset Manager	Managing Assets
KM	Knowledge Manager	Knowledge Management
RA	Reuse Administrator	Seeking for Reusable Parts

Table 6.3: *Roles in ISO/IEC 12207 (adapted from [3, 46])*

The third group has the organizational roles (see Table 6.3), the *manager* identifies and manages the state of the play (i.e. condition and progression of the project) with respect to project constraints (e.g. objectives, budget, schedules), the *asset manager* is a type of manager who deals with the management and optimization of the assets regarding the plan he or she prepared, the *knowledge manager* works on the collection of particular knowledge and skills throughout the organization and uses this for the improvement of the products and services. The *reuse program administrator* seeks to find favorable or advantageous circumstances for reusable parts of a product or a service. Unlike the other two subfields of software engineering (i.e. requirements engineering and software development), *domain engineer* is a form responsible for designing the domain models (i.e. software models) and domain descriptions for a software system.

6.2.4 Roles in Extreme Programming

According to Beck [58], the participants and their roles are as follows (see table 6.4); *Programmers* are the individuals who need to have good communication and collaboration skills for both team and individual levels. They are responsible for developing, maintaining and testing the software. One of their main responsibilities is to ensure that their work is clean and lean. The programmers make the technical decisions. *Customers* form the steering teams in business terms and in particular in requirement satisfaction decisions. *Testers* help customers to write functional test cases. Business decisions are made by customers [58]. The *tracker* role composes a trace and feedback mechanism in XP. The estimations, goals and iterations made by teams are controlled by a tracker, who provides feedback. The *tracker* is also responsible for measuring constraints such as scarce resources and delivery times versus goal evaluation. The *coach* is accountable for XP project, who needs to understand the problems occurring during the process to instruct team members and transfer the information or sometimes experience among teams and individuals. Finally, the *manager* is responsible for final decisions, and also one aim of this role is to recognize problems likely occur during the development life-cycle.

Code	**Role Name**	**Primary Type of Value**
PRG	Programmers	Maintaining and Testing Software
CU	Customers	Managing Business Decisions
TS	Testers	Helps Costumers for Functional Test Cases
TRC	Tracker	Feedbacks and Estimations
CO	Coach	Supervise Team
CON	Consultant	Guides the Team for Problem Solving
MAN	Manager	Management

Table 6.4: *Roles in XP (adapted from [58, 270])*

6.2.5 Roles in Scrum

Schwaber and Beedle [59] single out six roles for the participants of Scrum (see Table 6.5). The *Scrum Master* is a type of management role specific to Scrum, who is responsible for the alignment of practices and rules, as they have organized. This role interacts not only with project team but also customer and management. Its aim is to maximize productivity by practicing the agile and

scrum values and monitoring the team to avoid any kind of complications. The *Product Owner* is responsible for exercising the project management and control activities. Additionally, he is also responsible for transforming the product backlog into product features. *Scrum Team* should be considered as a self-organizing team to produce a working piece of a product, where the team's main goal is to achieve time targeted objectives of each sprint. The *customer* will continuously evaluate the backlog items, and helps the selection for a sprint. The *management* is responsible for implementing the proper standards for the software development process. Additionally, this role encompasses decision-making activities and finalizing them at different stages of development process such as evaluating goals, gathering requirements, etc.

Code	**Role Name**	**Primary Type of Value**
SM	Scrum Master	Managing Scrum Team
PO	Product Owner	Product Management Decisions
CUS	Customer	Evaluation of backlog items
STM	Scrum Team	Organized itself for time boxed goals
MNG	Management	Evaluate Decisions and Goals
USR	User	Evaluate System Functionalities

Table 6.5: *Roles in SCRUM (adapted from [59])*

6.2.6 Roles in FDD

FDD has the most comprehensive role description via flexibility of roles [60] (see table 6.6). For example, an individual can play multiple roles, or a role can be shared by multiple persons [270]. The three main categories of roles: *key, supporting and additional* roles. The key roles are *project manager*, who administers the entire project and maintains the work settings of the software team, the *lead software architect*, who makes the appropriate decisions for software development, and the *software development manager*, who focuses on daily activities and team negotiations during the software development activities. The *lead programmer, the class owner and the domain expert* are the three roles used in FDD. The supporting roles include *manager (release), knowledge expert, build process engineer, toolsmith* and *system administrator*. Moreover, *testers*, *technical document expert* and *software deployment personnel* are the other roles used in these practices [60].

Code	Role Name	Primary Type of Value
FPM	Project Manager	Resource Management
LSA	Lead Software Architect	Architectural Decisions
DEM	Development Manager	Evaluation of backlog items
LP	Lead Programmer	Organized itself for time boxed goals
CLO	Class Owner	Form Teams for Implementing Features
DE	Domain Expert	Inform Teams for Adequate Features
RM	Release Manager	Managing the development process
DM	Domain Manager	Managing Domain Experts
LG	Language Guru	Acquiring a Knowledge on Technology
BE	Build Engineer	Executing a Build Process
TO	Toolsmith	Creating Utilities for project
SYA	System Administrator	Administration of Work Systems
TE	Testing	Verifying the Actualization of a System
DEP	Deployer	Release of Feature Deployment
TEW	Technical Writer	The Documentation for Users

Table 6.6: *Roles in FDD (adapted from [60, 270])*

6.2.7 Roles in People CMM

The People CMM (PCMM) [271] is an organizational change management system that operationalizes the capabilities of the workforce by using the actor and role based elements. It suggests practices that are not specific to organizations and therefore PCMM implementations may have a different selection of roles specific to a software organization. There are several roles for the individuals that are responsible for organizations workforce activities identified as follows: (i) *executive managers* who are responsible for long term goals and preserving resources for long term improvements, (ii) *managers* who are managing individuals during their activities regarding to their area of authority, (iii) *individuals and workforce* (i.e. a set of individuals) who are responsible for the roles that are assigned with respect to business plans where there is no limitation for any individual to perform more than one task or role [272].

In addition, PCMM has five maturity levels. At level two, the roles inside the organization have responsibilities within the process areas. Starting at level three, individuals with more responsibilities emerge such as *process owners* or *competency managers* who may have broader authorities whereas the role called *human resource function* is responsible for recruiting, hiring, training, coordinating activities of the workforce, and regulate the relationships among the individuals so as to improve the organizational values [272]. Although PCMM is the only model that aims to align the capabilities of workforce and

related roles with the organizational human resources, in the model there is no visible association between these elements and the role descriptions and its characteristics, which are informally represented [3].

6.2.8 A Summary of Roles Contained in Selected Models

In this chapter, we highlight how roles in literature and their actualizations on industrial environments vary for both traditional and agile methodologies. Software development is a collaborative endeavor that depends on its development methodology. However, selection of a proper methodology is not enough for achieving goals of a software organization. The evidence suggests that we should also tailor the necessary roles depending on development activities.
After analyzing the defined categories in light of the questions above, we confirmed that several roles presented in traditional methods are emerged with a different name, with similar responsibilities in newer approaches. Some of the roles, however, have their responsibilities changed while implementing in different software development organizations. In addition, we introduce the role-job description sets, which identifies how a job fits to the role structure of a software development organization.

		Role-Job Descriptions			
		Actor Based	Activity Based	Artifact Based	Extended
Models	Traditional		✓		
	System Engineering		✓		✓
	ISO/IEC 12207		✓		
	XP	✓	✓	✓	
	Scrum		✓	✓	
	FDD	✓	✓	✓	✓
	People CMM	✓	✓		✓

Table 6.7: *Comparison of Role-job Descriptions*

Here, we present role-job descriptions for the selected software development methodologies as shown in Table 6.7. We identify four types of role-job descriptions: Actor-based, activity-based, artifact-based and methodologies with extended role definitions based on a previously defined role. For example, both scrum and FDD have actor-based roles, in which the skills of an individual are defined by the role characteristics such as product owner or a class owner. In addition, all methodologies have activity-based roles such as a software devel-

oper or a software tester. We also consider roles that are based on a creation of an artifact, which are highlighted by the agile methodologies. Finally, extended roles are the roles that can be integrated or shared among the individuals such as the roles like the domain expert role, which somehow comprises the technical writer role in FDD.

6.2.9 The Roles Wheel

Our analysis exhibits that a role-based schema can be useful for a tailoring process of roles regarding the organizational needs. Furthermore, we argue that a software development organization should customize their own roles to be suitable for their social structure, where we suggest that our role based construct (see Figure 6.1) will be beneficial for such activities. In other words, it enables them to select appropriate roles for their software development methodologies. Consequently, by using such a framework, a software team may easily choose or customize the necessary roles based on their activities.

The analysis of identified roles from the literature is portrayed in Figure 6.1. It is evident that several roles presented in older methods emerge with a different name with similar responsibilities in newer approaches. The roles, however, mostly have their responsibilities changed and reappeared as another form while revealing in different software development organizations. Most frequently, the role definitions that an organization uses should be based on a domain and a set of circumstances. Moreover, we suggest that the role selection should be based on the social structure of an organization and required interactions. Ultimately, the customized roles are found to be organizational centric, which also clearly supports the notion of *separation of concerns* [201].

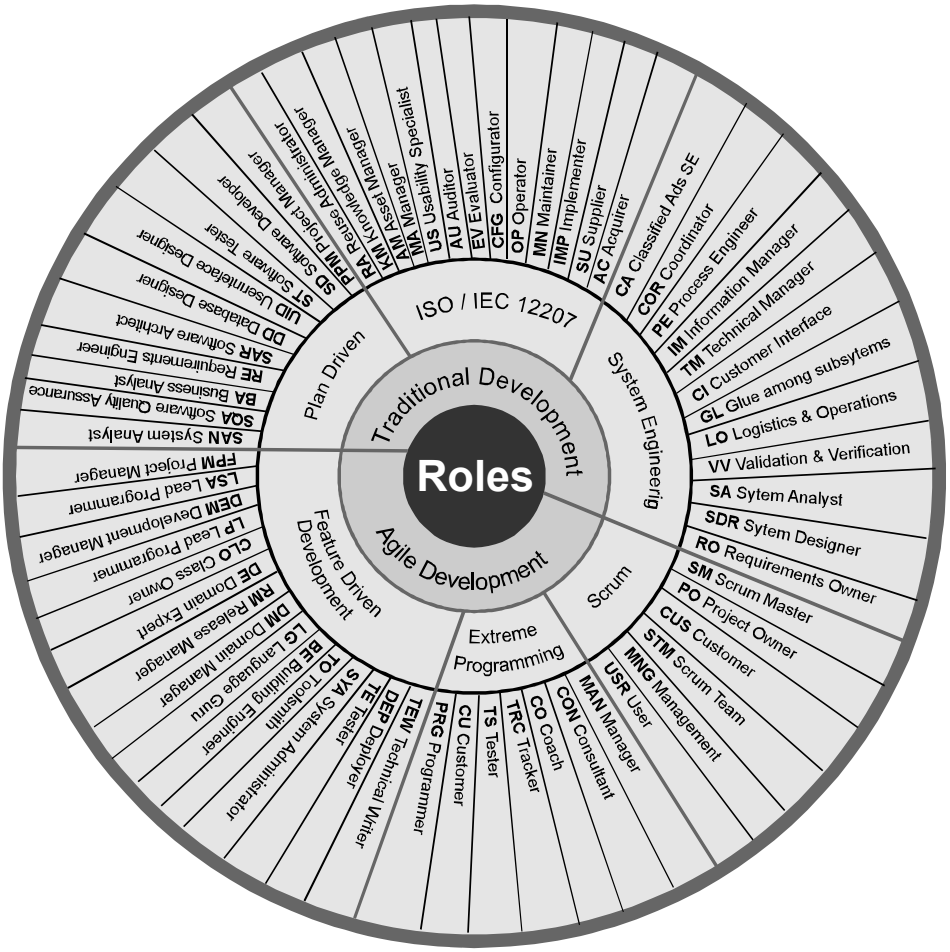


Figure 6.1: *A Summary of Roles Contained in the Different Approaches*

6.3 Personality Research

The field of psychology offers numerous definitions of personality. Although different opinions still exist, there appears to be some agreement that the notion of personality refers to all those characteristics that make every person a distinctive individual. Although the dimensions of personality are found to be dynamic and evolve over a period of time,"... there is a core of consistency which defines the individual's *true nature*: the unchangeable spots of the leopard. In other words, there are differences between individuals that are apparent

across a variety of situations" [36, pp. 37]. According to American Psychiatric Association, personality is "*...enduring patterns of perceiving, relating to, and thinking about the environment and oneself that are exhibited in a wide range of social and personal contexts*" [273, pp. 686].

A theory of personality types was introduced by Jung [274], who has established the classification of people with an orthogonal set of characteristics, which reveals individual's personality differences. The goal is to understand how and why individuals process a situation or an event and further function in both mental and emotional aspects represented by the traits. Based on the theory of Jung, Kathrine Myers and her daughter Isabel Briggs designed a questionnaire called Myers-Briggs Type Indicator (MBTI). Different from Jung's initial pairs, they suggested a novel attitude pair termed as perception versus judgment. The aim is to operationalize Jung's work in order to categorize people in terms of their personalities into 16 different psychological types, where the types were considered a combination of four preferences [275, 276]. Moreover, Myers and Briggs define these types based on the four preference pairs (EI, SN, TF, JP) or sometimes called dichotomies, namely; extroversion-introversion, sensing-intuition, thinking-feeling, and judging-perceiving, as illustrated in Table 6.8. According to the theory, each person has one preference from each of these bipolar factors and uses four out of the eight preferences (e.g. ISTJ, INTP, etc.).

Extroversion (E)	(I) Introversion
Sensing (S)	(N) iNtuition
Thinking (T)	(F) Feeling
Judgment (J)	(P) Perception

Table 6.8: *Dichotomies (the four opposite pairs of preferences)*

Despite the opposing views, some researchers argue that the psychometric properties of the identified pairs of preferences lack statistical validity [39], which may not be used as the absolute personality measure but useful to view the traits in a continuum [277]. From an industrial perspective, however, it is still one of the most widely used personality test [278].

There are several differences proposed to identify these dichotomous pairs. For

instance, the notion of being either extroverted (E) or introverted (I) shows the different preference of people and particularly the types (direction) of energy they have such as enjoying parties, meeting with new people or otherwise activities like reading, etc. Furthermore, socio-type (E) indicates the individuals who posses a positive altitude to social learning and prefer to have interactions to regain their energy whereas (I) types are the individuals who are seekers of knowledge and connections among them, thus individuals who prefer interconnected ideas of concepts and abstractions. Sensing (S) and Intuition (N) is the preference to process the data. (S) type prefers organized details and observable facts over the imagination to visualize. In contrast, (N) people are interested in the future and trust their gut feelings. Thinking (T) and feeling (F) dichotomy deals with how an individual makes decisions. The socio-type (T) seeks to observe cause and effect and logical sequences where justice and decency is more important than the other factors. On the other hand, (F) type is more personal, and less objective, so inclined to align herself with people of oriented concerns and their value dynamics since they enjoy in working social groups and its harmony. Finally, the preference between (J) and (P) dichotomies shows the path in which individuals view the life; whether one could prefer to be conclusive and systematically organized and deadline driven or otherwise more instinctive, and less organized but may be more adaptable to changes.

6.3.1 Jung's Model of Cognitive Modes

Earlier research suggests that personality traits encompass patterns of action in different situations, which should also need to have features like adaptability to the environment when needed. In his book *Personality Theory*, Jung [274] claimed that *attitudes* and *functions* of consciousness should be differentiated. In general, a decision process can be characterized by two actions; (i) retrieving the information from the environment, and (ii) making a decision based on this information. During these activities individuals may evaluate information by either their own memory and intellect, which is called *introverted perception* (Pi) functions or otherwise by equating a collective standard as seen in *extro-*

verted perception (Pe). Accordingly, for judging trait, there are also introverted and extroverted viewpoints. Individuals who perform *introverted judging* (Ji) usually compare their decisions by their own intellectual knowledge bank. By contrast, individuals with *extroverted judging* (Je) examine decisions based on the norms or the rules that are previously established. These functions are considered to work synchronously for the process of decision-making for every individual, although some might perform better for than others.

Table 6.9 outlines Jung's cognitive Modes for both information collection and decision-making processes of individuals adapted from [279].

Extroverted *Sensing* **Experimentation**	Extroverted *iNtuition* **Ideation**	Extroverted *Thinking* **Organization**	Extroverted *Feeling* **Community**
Introverted *Sensing* **Knowledge**	Introverted *iNtuition* **Imagination**	Introverted *Thinking* **Analysis**	Introverted *Feeling* **Evaluation**

Table 6.9: *Jung's Cognitive Modes*

6.3.2 Personality Research in Software Engineering

A considerable amount of literature has been published on personality research in software engineering where most of these studies were conducted using the MBTI [280]. A number of other instruments were designed to assess personality traits such as Keirsey [281] temperament sorter, which uses an MBTI compatible scale.

Preliminary work on the impact of personality traits on software team structures was undertaken by White [282], who reported that the diversity in personality traits were beneficial for dealing with a number of software development activities. Kaiser and Bostrom [283] conducted a study, which confirmed that personality types have a significant impact on the success of a management information systems team. They hypothesized that successful project teams usually include a variety of personality types. Consequently, it was claimed that the absence of feeling (F) personality type in a team directly affects the project success. By using Keirsey temperament sorter, Rutherford [284] conducted a study to build project teams for software engineering classes. Likewise,

he concluded that a team with a variety of personalities had more skills in problem solving. In addition, based on the MBTI scale in their ethnographic study Karn and Cowling [285] conducted a performance analysis of student teams by comparing team effectiveness in yearly basis. Their findings also suggest that building heterogeneous teams in terms of individual personalities brought different ideas to teamwork, which improved the team's productivity.

To improve pair forming process and its effectiveness, Sfetsos et al. [286] empirically investigated the effect of MBTI personalities and Keirsey's temperaments over novice (student) developers in pair programming. Their study compiled with the idea that pairs with diverse personalities were more effective than homogeneous team of pairs. Based on the MBTI personality traits, Dick and Zarnett [287] claimed that pair programming was only suitable for a limited number of people, and therefore a preference should be set based on the traits of individuals in a team. In addition, Karn et al. [288] conducted a qualitative analysis to investigate how dynamics of software teams can be related with personality type research, particularly for XP projects. The results of this analysis indicated that (i) personality type configurations were important for team effectiveness, (ii) teams with high cohesion were found more competitive.

Prior studies that have noted the importance of personality research in software engineering point out that the socio-type *ISTJ* was found as the most frequent trait in this particular domain. Bush and Schkade [289] found 25% of scientific programmers are ISTJ, where Buie [290] singled out 19% of programmers, and further Smith [291] found 35% of system analyst were *ISTJ*. A two phased MBTI based study by Turley and Bieman [292] reported that the programmers in their small empirical sample were mostly found to be introverted (I) and thinking (T) type.

From an industrial point of view, Hardiman [293] investigated the traits of software engineers who were found to be *ENTJ, INTJ, ESTJ, ISTJ, ISFJ*, and *ENTP*. He claimed that individuals with *NF* trait suffered from a lack of process-based thinking. However, a limitation of this study was that the numbers of participants were relatively small. Carpetz [294] surveyed software

engineering students by using an MBTI scale, which also found that introverted and thinking types are more than extroverts. In this work, the number of sociotype *ISTJ* was also found more than the other types (24%). The details about the results of *ISTJ* dominant surveys were also summarized in [294]. Moreover, Carpetz concluded that a variety of personality types should form better teams to improve the quality of products.

Sach et al. [295] analyzed five different studies conducted to investigate MBTI preferences in the software engineering domain where they found that these studies supported each other. In general, for example, thinkers (T) were commonly higher than feeler (F) type. Detailed examination of the relationship between personality types versus variance of performance in code review by Da Cunha and Greathead [296], who proposed that a productivity difference among the individuals might emerge when teams were organized according to their personality types. In contrast with some of the previous findings, their study reported that only 6% of their sample set was *ISTJ*.

In a study in 2004, Gorla and Lam [297] reported personality traits of 92 personnel practitioners that were structured in small teams using Keirsey's temperament sorter. Their goal was to find a possible connection between personality traits of teams and their performances. Not surprisingly perhaps, it was found that extroverted individuals communicate better than an introverted personnel, therefore extroversion was a preferable type of personality particularly for tasks that requires more social interactions. From a traditional viewpoint, they argued that personality traits known as sensing (S), and judging (J) are more suitable as programmers' characteristics.

In recent years, there has been an increasing amount of literature reporting that the personality of software development practitioners known as *less socially interactive type* begin to change due to several reasons such as a set of roles that emerged with different social requirement for success, e.g. system analyst, interface designers or software testers, etc. To investigate the personality types of Cuban software developers, Varona et al. [298] surveyed 103 individuals where the analysis reveals that one of the most conspicuous personality characteris-

tics was *ESTJ*. In accordance with this, one of the recent findings by Varona et al. [299] points out a significant change in preference, from traditional introverted software engineers to the extroverted trait where they also identify that people with more social skills were needed to deal with the complexities of new software projects.

Taken together, an implication of previous research on personality traits demonstrates that it is important to reveal the personality traits in a software team [300]. Such a team with a variety of different skilled people with distinctive perspectives should be beneficial to cope with the dynamic tasks of a software development process [301]. For instance, some of the personality traits such as introversion vs. extroversion may have more impact on one development phase, and can be used for selecting individuals for the different stages of a software development process with respect to their so-called *soft skills* [302]. As a consequence of the growing complexity of software development activities, there is no single personality trait found that fulfills all roles in software development. Moreover, an ethnographic study targeting the actualization of MBTI measures on software teams by Karn and Cowling [303] reached a conclusion that specific personality traits are more gravitated towards some specific type of development activities, as well as the roles.

From an industrial perspective, MBTI is considered as a useful tool in the service of exploring the social characteristics of the software practitioners during the activities of software development [40]; however, there is no systematic and rigorous method to relate the personality traits of software practitioners to the formation of a software team structure in the literature. Kaluzniacky [40] agrees with this pointing out that a tool for assessing the personality characteristics of IT practitioners should be constructed. In addition, most of the research conducted in MBTI literature is qualitative, which means that the research to date has tended to focus on individuals who can either be found extroverted or introverted without any quantification such as percentage of extroversion level. This study aims to measure the dichotomies of personalities on a quantitative scale, which can be useful particularly for illustrating the personality

characteristics on team level.

6.3.3 Personality Temperaments

As defined in the previous section, the term personality tends to be used to refer to a set of characteristics based on the thoughts, emotional and behavioral patterns that constitute our talents, emotions, habits, and skills, which are specific to an individual. The term *temperament*, however, is generally understood to mean a set of personality traits inherited such as intelligence and the acquired dispositions such as background, social and cultural history, etc. [304]. Allport defines temperament as a component that refers to emotional nature of individuals [305], e.g. sensitivity to provocation, responsiveness, ability to keep the intensity of mood, etc. Therefore, the notion of temperament can be considered as an *intangible substratum* (i.e. a basis or a foundation) for a consequent advancement of personality characteristics based on various life experiences that affect an individual's basic inclinations [306].
For Keirsey, temperament is of four kinds [281]:

- Stabilizers (*Guardians*): This temperament can be found in cooperative traditionalists, who values protection and stability the most; they prefer to be part of an organization, and like to live by the rules. Mostly, they are found to be reliable and hard working individuals.

- Improvisers (*Artisans*): They are equipped with advanced tactical skills, and are the most talented group of individuals on using a tool such as software, screwdriver, language, etc. They are mostly realistic, sometimes unconventional, freedom lover, and their favorite expression is *carpe diem*, i.e. seize the day.

- Catalyst (*Idealists*): The people in this temperament are known to be the most communicative type. They are politic, enthusiastic, intuitive, sometimes more romantic and spiritual. They love gaining knowledge and self-improvement, and also like to guide individuals on these kinds of quests.

- Rationalists (*Theorists*): This group of people are famous with their logic and problem solving skills. Mostly, they are skeptic, pragmatic, and independent. They don't prefer to work on understanding how things work. They are not good at diplomacy, and sometimes not in digesting details.

Figure 6.2 illustrates the relationship between Keirsey's temperaments and MBTI personality types adopted from [281]. Keirsey's personality classification starts with understanding the way an individual responds the world either perceiving concrete or abstract realities. The individuals' several concerns that correspond the four key characteristics can be listed as follows: (i) guardians are the concrete cooperators who represent the logistic intelligence, and the ability to differentiate the problems using logistic interpretations, (ii) artisans are the concrete utilitarians who have the tactical intelligence which helps them to aim for shorter time-frames to achieve goals, (iii) idealists are the abstract cooperators with diplomatic intelligence who have the skills to decide the factors for bargaining, and lastly (iv) rationalists are the abstract utilitarians who have the strategic intelligence, which helps them to achieve long term objectives [307].

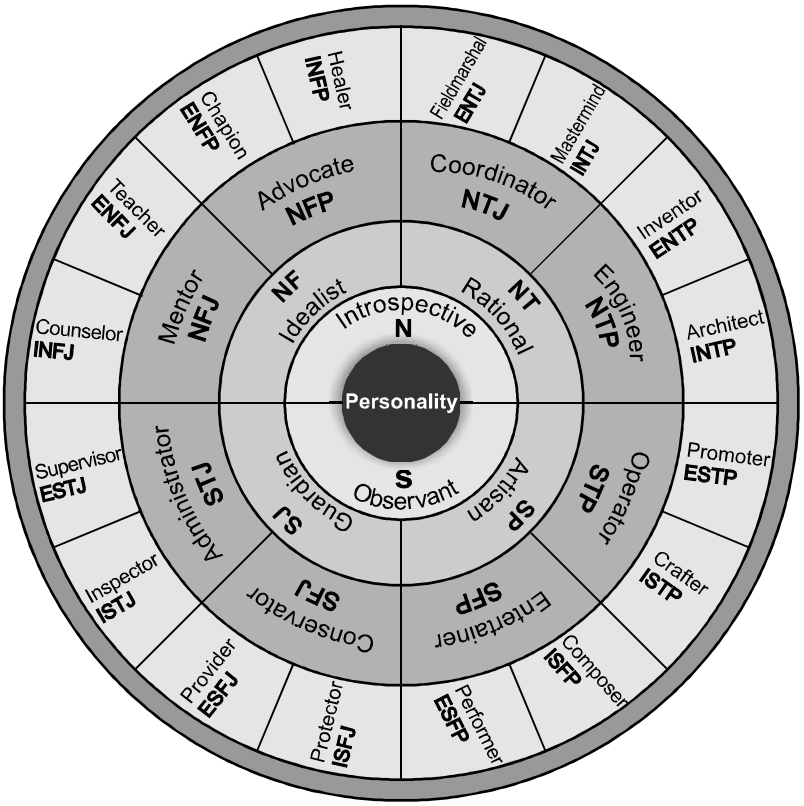


Figure 6.2: *The Personality Wheel*

6.3.4 The Periodic Table Approach

The *abstract notion* of a periodic approach relies on the fact that there is an axiomatic relationship among a group of entities [308]. However, common form of a periodic table comprises rows and columns in which the classification is made by the entities placed across or down the table. For the purpose of studying the interpretations of the individuals in software teams based on their personality traits, we compile a periodic table-like structure. In the context of this study, a periodic table system is a compilation of the characteristics in a compact form for classifying sixteen forms of personality traits. Basically, it is a tabular depiction of the personality traits (see Figure 6.3), which is organized with respect to a set of commonalities among the personality characteristics.
To facilitate the study of the relationships of personality traits, we propose a periodic table-like structure. In this type of representation, commonly, rows

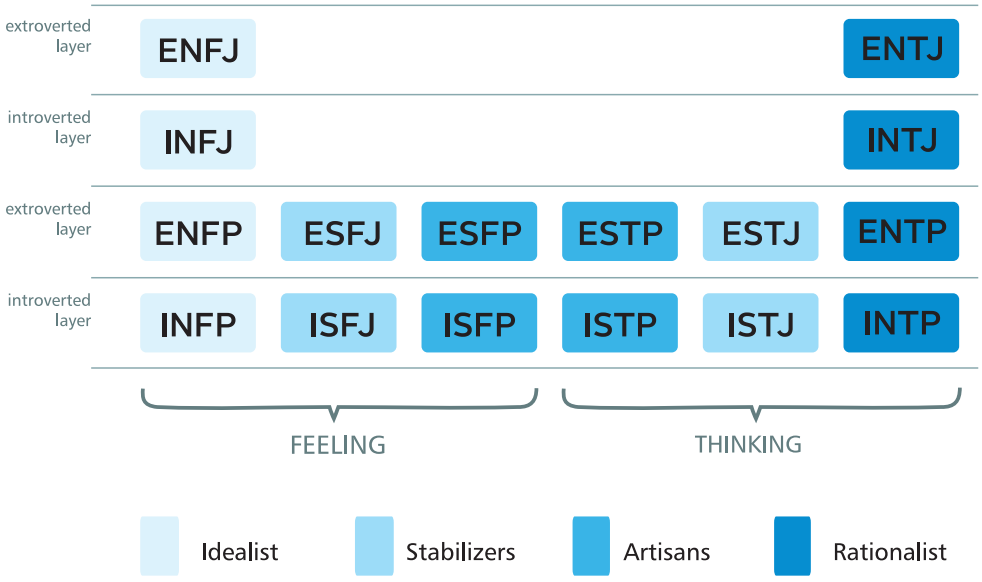


Figure 6.3: *The Periodic Table Type Classification for Personality Types*

and columns represent a classification with different attitudes whereas the opposite ends of a continuum are based on the features of the taxonomy. The traits horizontally are divided into two layers based on the social attributes, i.e. a row represents either extroversion and introversion group layer. The vertical columns on both sides of the table designate the level of rationality and emotionality of the traits, i.e. altruistic through individualistic. The vertical columns inside the table are formed regarding to individual's negation and social stabilization skills.

Table 6.10 provides a periodic table-based visualization of a team-based snapshot for a software organization in terms of practitioners' personality traits. The periodic table is employed as a team illustration format with which we can use to visualize all software teams inspected in our study.

ENFJ(%)					ENTJ(%)
INFJ(%)					INTJ(%)
ENFP(%)	ESFJ(%)	ESFP(%)	ESTP(%)	ESTJ(%)	ENTP(%)
INFP(%)	ISFJ(%)	ISFP(%)	ISTP(%)	ISTJ(%)	INTP(%)

Table 6.10: *Periodic Table of (%) Personality Traits of a Software Development Organization*

Bradbary and Garrett [309] indicate the benefits of using personality temperaments in software team management as follows:

> *"....Say you've got a problem that needs a novel solution. Assign an Inventor (ENTP) or a Crafter (ISTP) to handle the job. Both thrive on ingenious problem-solving; they're good with Gordian Knots. If, on the other hand, you've got a large, messy project that needs to be organized and whipped into shape, call on a Field Marshall (ENTJ). These little Napoleons know how to regiment people and resources alike (hence their name). Architects (INTP) are good at big, complex problems that need fine-tuning its the nature of their intellect to tweak and tinker. And Counselors (INFJ) have a talent for issues that need a touch of tact and empathy." [309, pp. 37].*

Table 6.11 illustrates a periodic table-based visualization of software teams, which shows the corresponding Keirsey Temperaments.

Teacher					Fieldmarshal
Counselor					Mastermind
Champion	Provider	Performer	Promoter	Supervisor	Inventor
Healer	Protector	Composer	Crafter	Inspector	Architect

Table 6.11: *Periodic Table Representation of Temperaments*

To sum up, the periodic table form gives a complete visual representation of MBTI based personality types classified by using the temperament information extracted from Keirsey's book [281], which demonstrates the benefits of external representations of practitioners in software teams. Ultimately, a goal is to investigate the relationships between the personality characteristics of team members and social structures of effective team configurations.

6.4 Chapter Summary

This chapter highlighted the importance of roles and personality traits in software engineering landscapes. In order to identify and compare different roles in software development activities, we built a framework for comparison of software development models, covering traditional approaches through to agile techniques. Furthermore, we built a comparison schema of roles for the selected development methodologies, which can be useful both in academic and industrial aspects. In particular, the approach can be beneficial when selecting the proper roles for a tailored software development process.

The latter parts of this chapter outlined the Jungian personality types, and continued with a literature review of personality research in software engineering. It concluded with a novel idea later we use in our empirical studies - a periodic table form. The goal was to visualize a team with respect to personality temperaments of its participants. What's more, it could be useful for assigning practitioners on software teams based on their personality characteristics, and providing feedback on the team dynamics. This also enables the participants in a software team to know more about the personality types of other members of that team.

In the following part of this thesis, we will detail the research processes and the first chapter is going to present the empirical analyses and their findings from the two industrial case studies.

Part IV

Empirical Contributions

Industrial Case Studies

This part of the thesis presents two industrial case studies conducted in a middle size software company. It outlines the descriptions about the population samples, the techniques and methods used for data processing and the analyses. Firstly, Chapter 7 introduces the industrial case study conducted for investigating the factors affecting software development productivity. Secondly, Chapter 8 discusses the findings of an industrial case study conducted for revealing the personality types of individuals for understanding effective team structures.

Chapter 7

Empirical Findings: Case Study I

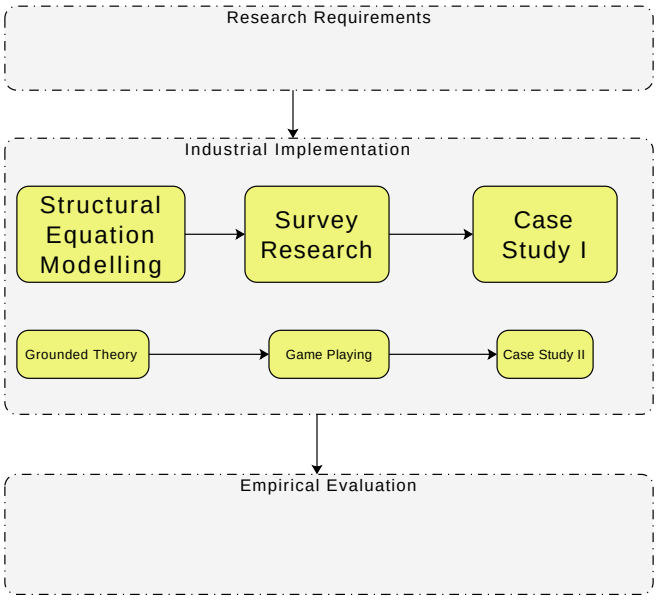


Figure 7.1: *A part of the conceptual overview of the research*

7.1 Introduction

This chapter aims to enhance our understanding of the factors affecting an organization's social and economic structure so as to improve the productivity of software development. We present an approach for identifying the many factors affecting the development productivity and enhance our understanding of

productivity as a latent construct. To demonstrate the relationship between observed indicators and latent constructs of productivity, we build and empirically test a series of structural equation models (SEM) with data collected from 216 participants from a medium-sized software company (see e.g. Figure 5.1). Secondly, researchers analyze the impact of a set of team-based variables for substantial productivity improvements. In light of these, several important factors found in the literature in Chapter 4 are tested for measuring the relationships among the latent constructs.

7.2 Data Collection

After discussing our research objectives with a number of companies, we selected a middle-sized software development organization, *Simurg*[1]. The first reason was that they were willing to participate in the research. Secondly, the company employs more than four hundred people. Therefore, the size was adequate, likely to increase the reliability of outputs. What is more, the management group of Simurg was interested in the factors affecting their productivity, and therefore they were willing to contribute to the study with more accurate findings. Consequently, survey questions were developed from the factors found in the literature, and their content validity was reviewed by the management team. Additionally, Simurg is an organization in which development activities occur in multiple locations, and the number of software teams vary between four to forty members. This would increase our chances of for observing different team configurations especially required for the second part of this study.

To evaluate our hypothetical model with the empirical data, we developed a survey instrument based on two different resources: (i) literature review of the factors of productivity, social productivity, and social capital affecting a software development organization, (ii) focus group study conducted with the management team. Furthermore, we examine a set of the documentation and a series of case reports previously prepared about organizational productivity.

[1]To protect the identity of the firm, we will use a fictitious name.

7.2.1 Industrial Focus Group

After having chosen factors of both productivity and social productivity, a focus group study was conducted to investigate the opinions of software management teams in Simurg. The goal of the focus group study was to identify the opinions from industry about the most important factors that are affecting productivity. The discussion group was composed of nine personnel from the management team (total ten participants). As suggested by Krueger [310], the session was facilitated by one of the authors who commenced an introduction to encourage participants and initiate the discussion setting. We asked the management team about their opinion on productivity factors and one individual from the management team took notes. A guide containing five questions and a preliminary model of social productivity was prepared for the focus group discussion: (1) What is your definition of productivity in software teams?, (2) What is your opinion of the factors that are affecting the productivity?, (3) Which one do you think is the most important factor among these ones for productivity?, (4) How would you describe the social factors of productivity?, (5) What social factors do you think are affecting the productivity?

The participants discussed the social aspects of productivity including the impacts of social values over productivity, the communication frequency, coordination efficiency, team augmentation, and task rotation. In addition, the group discussed the selected items from the software productivity literature. In short, this focus group activity provided us with an opportunity to discuss our ideas about productivity factors in an industrial setting. We refined our list of factors found from the literature by using the information provided in this session.

7.2.2 Survey Instrument

The questionnaire had questions about the potential factors from literature of software development productivity using a 5-point Likert scale grading between strongly agree (5) and strongly disagree (1) for productivity (see Appendix A). Additionally, the survey had several questions like gender, years of work experience of the participants in this company as well as the ideal team size and

the actual size of their software team. The survey questions were developed to measure the relationships between the observable factors and latent constructs of productivity. To ensure proper interpretation of each question, we worked with several experts from Simurg's management team. In light of these efforts, a set of survey questions were constructed, and refined. Ultimately, the management team of Simurg announced the finalized version of the survey at their internal web-portal.

To increase accessibility, a variant of the questionnaire was prepared with the LimeSurvey. It is an open source web-based tool for conducting surveys, which was employed as the primary instrument of data collection. The survey ran approximately for one month, which has 57 questions and also has an introductory letter and confidentiality statement. To increase the understandability of questions, we built five question categories that were presented in saveable web-based sessions. The five categorical parts represent five different aspects that we were investigating. It was also used as a stopover for participants and to store their answers if need be. Although nearly all members of the company are bilingual, our survey was available in both Turkish and English with the following parts: (i) 17 questions about the factors affecting software productivity elicited from the literature such as motivation, management quality (e.g. process, development tools, programming languages), complexity issues (e.g. task, process, product), work environment, re-usability, requirements stability, team issues (e.g. size, organization, location); (ii) 12 questions about the social productivity factors identified from the literature such as conflicts and reputation of a team leader, social interaction, social life, information awareness, team cohesion, fairness, frequent meetings, and social trust; (iii) 10 questions about the factors of social capital surveyed in the literature such as neighborhood connections, group characteristics (personality types), generalized norms, togetherness, everyday sociability, volunteerism, and trust; (iv) 12 personality type questions to determine the importance of factors affecting the personality traits based on the personality type constructs, and (v) 6 other complementary and control questions (see Appendix C for survey data).

7.3 Data Analysis

To test our hypothetical model of productivity and to reveal the relationships of the latent and observable factors that are selected from the literature, we perform a structural model analysis by using the linear and continuous framework of LISREL [311], which is one of the most popular computer tool for SEM analysis. In order to build a measurement (factor analysis) model, we attempt to measure the unobserved variables (i.e. latent constructs) by using the relationships of each observed variable with a construct.

Of the initial cohort of 213 industrial participants who returned the questionnaire, 21 were excluded as their questionnaires had missing pairs. We ended up with 192 *appropriate* observations (cases) 24% of which were female, and 76% of which were male participants. Prior to data analysis, the Turkish translation of the survey was checked for both consistency and the language by a number of experts from industry and academia. Next, we analyzed the role distribution of the sample from the company Simurg. Table 7.1 shows the initial results.

Role[1]	**Number of Individuals**	**Percentage (%) in Organization**
IT Specialist	25	13
Project Manager	17	9
Software Architect	4	2
Software Developer	66	35
Team Leader	13	7
Software Tester	23	12
Software Specialist	29	15
System Analyst	10	5
System Engineer	5	2
Total	192	100

[1]See Figure 6.1 for a summary of roles in software development.

Table 7.1: *Distribution of Roles of the Participants in Development Organization*

To asses the *internal consistency* of the survey, we use Cronbach's α, a frequently used variable to measure the validity and reliability of responses collected by psychometric instruments [312]. The values around .70 or higher are reliable, where a high Cronbach's α value signifies that there are highly correlated variables that are found in the survey [313].

$$\alpha_{Cronbach} = \frac{N}{N-1}\left(\frac{S^2 - \sum S_i^2}{S^2}\right) \tag{7.1}$$

where N is the number of items in the questionnaire, S^2 is the *variance of total score* for each participants, $\sum S_i^2$ is the summation of variances for each question. Depending upon what is evaluated, the number of respondents or the number of questions is shown by j and the variance is calculated as follows:

$$S^2 = \frac{1}{j-1}\left(\sum_{i=1}^{j}(X_i - \bar{X})^2\right) \tag{7.2}$$

Table 7.2 below illustrates the Cronbach's α values for our survey instrument. Overall, it was apparent from our calculations that the responses to our survey had a high Cronbach's α value, .83, which means that the questionnaire is able to measure the latent constructs. In addition, we checked the consistency of each set of questions for the constructs of the survey. The important result to emerge from these calculations was that our survey instrument had an adequate consistency according to Cronbach's α values calculated for each of the selected constructs (see Appendix D for a sample calculation).

	Survey Constructs (Overall Cronbach's α = .83)		
	Productivity	Social Productivity	Social Capital
Cronbach's α values	.68	.73	.76

Table 7.2: *Individual Sections of the Questionnaire with respect to Reliability Coefficients*

Table 7.3 presents responses for all identified factors, their descriptions, standard deviations, and the variances as descriptive statistics calculated for each factor that potentially affects the productivity of software development.

In response to the survey instrument, most of the questions indicated that nearly all the factors proposed to affect software development productivity measures had a rating higher than 3, which was considered as the middle point in a 5-point Likert scale. This ensures our survey questions are relevant to participants.

After the survey was closed, we conducted a series of interviews to understand the problematic items in the questionnaire. The question, *teams in different locations*, was not interpreted properly. Later we found that the term *location* was understood differently, e.g. in the same office or otherwise in the same

Factor ID	Descriptions	Mean	s.d.	Variance
X1	Level of individuals motivation	4.72	0.49	0.24
X2	Level of interests of individuals for their assigned tasks	4.56	0.66	0.44
X3	Development process or methodology	3.94	0.77	0.59
X4	Programming language	3.77	0.90	0.81
X5	The tools and technologies used	4.29	0.67	0.45
X6	Complex and challenging tasks	3.97	0.95	0.89
X7	Large and complex structured projects	3.37	0.89	0.80
X8	Tasks and their complex connections	3.80	0.74	0.55
X9	The work environment	4.17	0.74	0.55
X10	Using an off-the-shelf product	3.80	0.97	0.93
X11	The ability of an organization to stabilize requirements	4.22	0.78	0.61
X12	The changes in requirements of a project	3.68	1.04	1.09
X13	The team size	3.64	1.02	1.05
X14	Verbal communication of team members	4.39	0.70	0.49
X15	Non-verbal communications	3.03	1.10	1.22
X16	Teams in different locations	2.33	1.10	1.21
X17	Internal problem solving skills of a team	4.19	0.76	0.58
X20	Team Leaders conflict resolution skills	3.74	0.91	0.82
X21	Team leaders general skills	4.42	0.59	0.35
X22	Communication with all team members	4.30	0.80	0.64
X23	Social life out of the work place	3.65	0.89	0.79
X24	Knowing the tasks of others	4.22	0.85	0.73
X25	Collective team memory	4.07	0.61	0.37
X26	The unity in the service of team goals	4.16	0.69	0.47
X27	Enjoying teammates company	3.81	1.04	1.09
X28	Working less than the others	3.50	1.17	1.36
X29	Fair allocation of work	4.08	0.73	0.54
X30	Frequent Meetings	3.96	0.95	0.90
X31	Social trust	4.29	0.67	0.45
X32	Social connections	3.64	0.97	0.93
X33	Efficient usage of the social connections	3.68	0.95	0.90
X34	Social connections and career success	3.39	1.00	0.99
X35	Variation of personalities	3.42	0.96	0.92
X36	Generalized norms	3.21	0.92	0.85
X37	Togetherness	2.21	1.05	1.10
X38	Everyday sociability	3.27	1.13	1.27
X39	Extra potion of work for more social connections	3.14	1.03	1.05
X40	Volunteerism	3.66	0.88	0.78
X41	Trust	3.84	0.81	0.66

Table 7.3: *Means, Variances and Standard Deviations of the Factors of Productivity*

country, etc. Similarly, our interviews revealed that *non-verbal communication* might also not interpreted as expected because it had different meanings for the participants. In general, therefore, the two problematic items were found and excluded from all models.

7.4 Confirmatory Factor Analysis and Construct Validity

The content of this section is concerned with estimating the relationships in the surveyed productivity factors with respect to each other. Structural modeling is therefore used for linking a theoretical perspective (i.e. proposed model) with the observed data. We formed several SEM models; first we built models based on all parameters mentioned in the survey data for productivity, social productivity and social capital. Secondly, we estimated the common factors and built a set of models to investigate the correlations between observed variables in specific cases. Finally, we built models to relate the constructs, productivity, social productivity, and social capital. To assess these constructs, we systematically conducted our analysis in three main steps: (i) testing the measurement model, (ii) testing the structural model, and finally (iii) comparing the model with the alternative models. The procedure, called the maximum likelihood estimation, was used to approximate the parameters of models with their loadings, i.e. the variables were hypothesized to be related with each other.[2]

7.5 Models with One Latent Construct

In this section, we hypothesized six models with one latent variable to test the validities of factors for latent constructs that are identified as productivity, social productivity, and social capital: The first one was designed (see 7.2) to analyze the factors of productivity. It employs all 17 observed variables identified from the literature, X1 through X17 (see Table 7.3). Structural correlations were statistically significant except for variables X15 and X16, namely the factors for the *teams in different locations*, one of which had a negative loading score and *non-verbal communications* was below the threshold (less than .20) [314], which is an indication of a comprehensibility problem particularly for these two questions.

We ran the analysis; good fit values were obtained where the indicators varied between .60 and .25 (see Figure 7.2). A null hypothesis (i.e. latent con-

[2]LISREL incorporates maximum likelihood estimation as the default procedure.

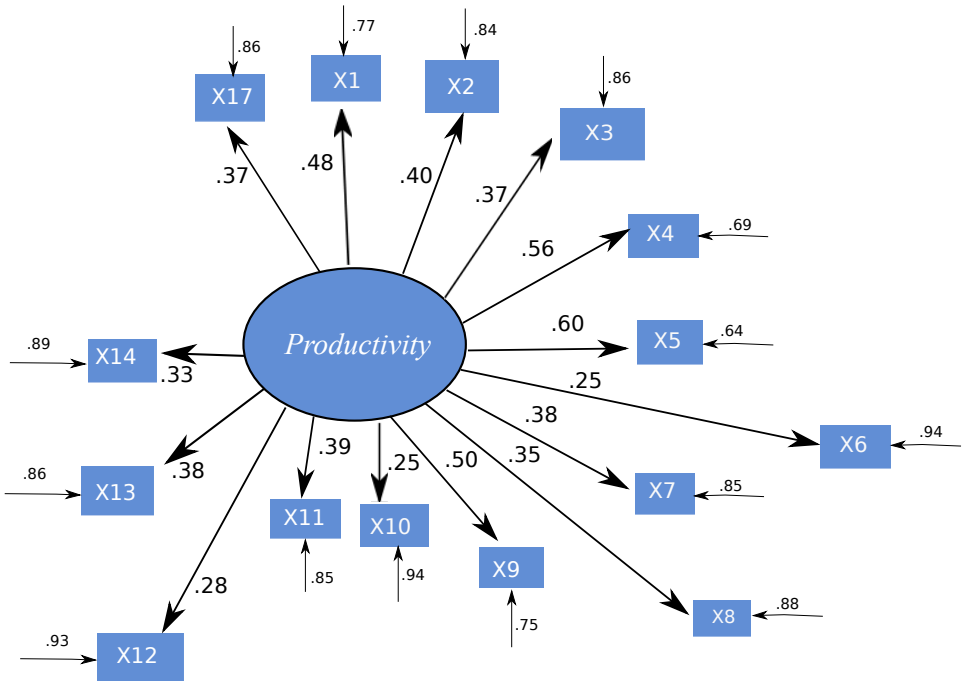


Figure 7.2: *Model I with Loadings with Fifteen Factors of Productivity of Software Development*

struct and variables are uncorrelated) was rejectable where $\chi^2(136, N = 192) = 684.053, p < .001$. In addition, there was a good-enough fit between the model and the data $\chi^2(119, N = 192) = 206.714, p < .001$, where $RMSEA = .0621, GFI = .887, AGFI = .855, CFI = .823, NNFI = .798$). A χ^2 difference test indicated that there was, however, a significant improvement between the independence model and the hypothesized model such that $\Delta\chi^2(17, N = 192) = 477.339, p < .001$).

In model II, we hypothesized that seven factors or variables that covary together are (i) level of individuals motivation, (ii) the tools and technologies used, (iii) tasks and their complex connections, (iv) the work environment, (v) use of off-the-shelf products, (vi) requirements stability (vii) problem solving skills of a team, shown by $X1, X5, X8, X9, X10, X11, X17$, respectively, as the model parameters (see Figure 7.3, model II).

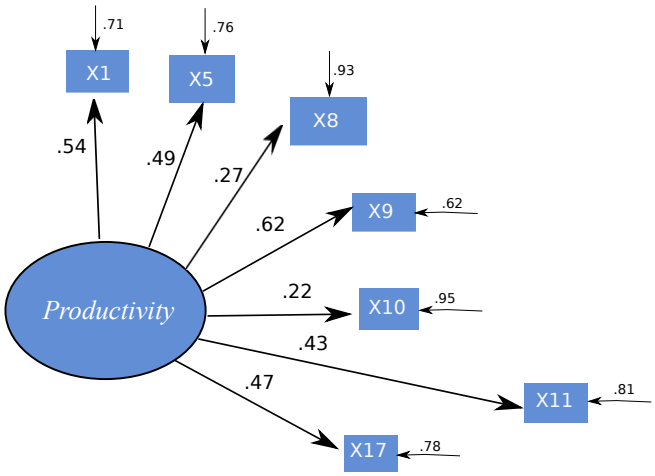


Figure 7.3: *Model II with Loadings with Company Selected Seven Factors of Productivity of Software Development*

Model II built for productivity has structural correlations that were statistically significant for all variables ($p < .05$). All factor loadings are above the threshold of .20 that ranged between .22 and .62. A null hypothesis was rejectable where χ^2 for independence model with 21 the degrees of freedom is 188.95. The model showed a moderate fit[3] with data, where $\chi^2(14, N = 192) = 38.110, (p < .001), RMSEA = .095, GFI = .89, AGFI = .90, CFI = .85, NNFI = .78)$. A χ^2 difference test indicated that there was a noticeable improvement between the independence model and the hypothesized model as $\Delta\chi^2(7, N = 192) = 150.84, (p < .001)$.

7.5.1 Models for Social Productivity

To investigate the relationship between the latent construct of social productivity and potential factors affecting it, we built multiple models with the surveyed factors affecting the social productivity construct. Consequently, the first model was based on the six factors, namely (i) team leaders' skills, (ii) communication among team members, (ii) social life (out of the work place), (iv) knowing the tasks of others, (v) fair allocation of work (vi) frequent meetings, ($X21, X22, X23, X24, X29, X30$, respectively) (see Figure 7.4).

[3]A moderate fit: " There are some differences between the model and the data but there are also some great similarities" [315]

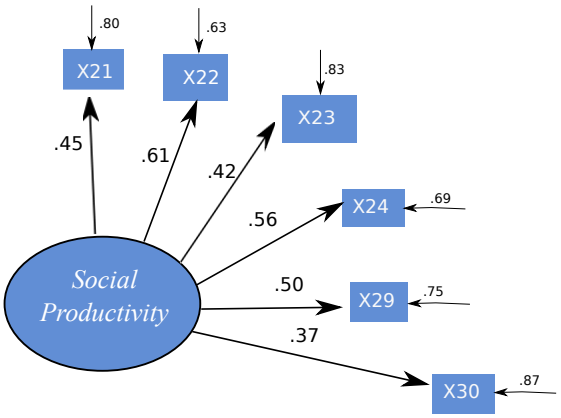


Figure 7.4: *Model III with Loadings for Six Factors of Social Productivity for Software Development*

The Model IV had eight hypothesized variables to define social productivity namely: (i) team leaders' skills, (ii) communication between the team members, (ii) social life out of the work place, (iv) knowing the tasks of others, (v) the unity in the service of team goals, (vi) fair allocation of work (vii) frequent meetings (ix) social trust, ($X21, X22, X23, X24, X26, X29, X30, X31$, respectively) (see Figure 7.5).

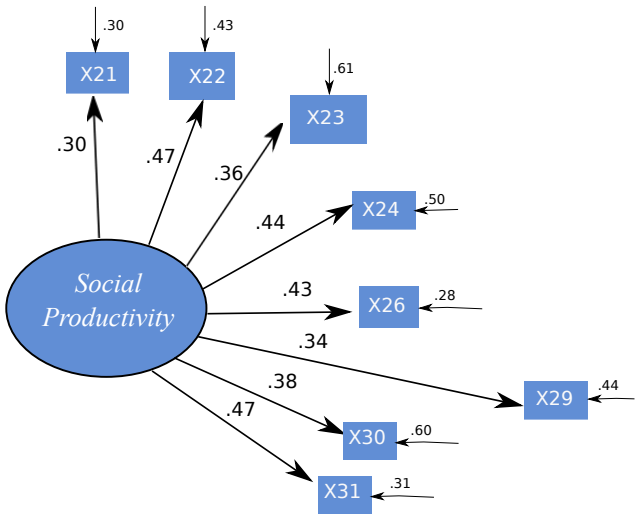


Figure 7.5: *Model IV with Loadings for Eight Company Selected Factors of Social Productivity for Software Development*

Both proposed social productivity models had structural correlations that were statistically significant ($p < .05$). For the third model (Figure 7.4), independence model was rejected; χ^2 for independence model with 21 Degrees

of Freedom is 174.242. All factor loadings are above the threshold of .20 where they ranged between .37 and .61. The proposed model showed a reasonable fit to the data, where $\chi^2(14, N = 192) = 23.792, p < .001$, and $RMSEA = .0622, GFI = .964, AGFI = .93, CFI = .934, NNFI = .901$. The difference test for χ^2 indicated that there was an improvement between null model and hypothesized one ($\Delta\chi^2(7, N = 192) = 150.45, p < .001$).

For the fourth model (Figure 7.5), the independence model with 28 degrees of freedom was 401.815, where the proposed model showed evidence of having a very good fit with the data, $\chi^2(20, N = 192) = 22.933, p < .001$, and $RMSEA = .0285, GFI = .969, AGFI = .945, CFI = .990, NNFI = .986$. All factor loadings were above the threshold of .20 where they ranged between .30 and .47. The difference test for χ^2 signified that there was an improvement in between null and the hypothesized model, ($\Delta\chi^2(8, N = 192) = 378.882, p < .001$).

7.5.2 Models for Social Capital

To investigate the relationship between the latent construct of social capital and potential factors affecting it, firstly, we built Model V as a social capital model (Figure 7.6) with seven observed variables by using the social capital model derived from work of Narayan and Cassidy [194], with the factors; (i) efficient usage of the social connections, (ii) social connections and career success, (iii) generalized norms, (iv) togetherness, (v) everyday sociability, (vi) volunteerism, (v) trust, ($X33, X34, X36, X37, X38, X40, X41$, respectively).

The first social capital model (Figure 7.6) had structural correlations that were statistically significant. All factor loadings were above the threshold of .20 where they ranged between .34 and .77. A null hypothesis was rejectable where χ^2 for independence model with 21 Degrees of Freedom was 274.796. The model showed a moderate fit with data, where $\chi^2(14, N = 192) = 34.703, p < .001$, and $RMSEA = .088, GFI = .951, AGFI = .901, CFI = .913, NNFI = .869$). A χ^2 difference test indicated that there was a significant advancement between the independence model and the hypothesized model as $\Delta\chi^2(7, N = 192) =$

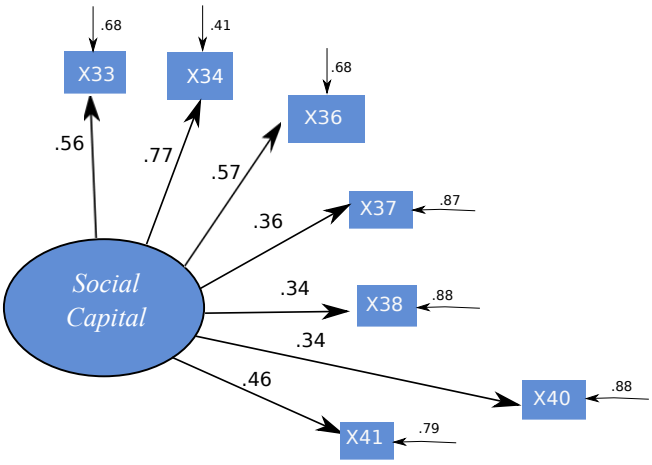


Figure 7.6: *Model V with Loadings with Seven Factors of Social Capital in a Software Development Organization*

240.093, ($p < .001$).

To investigate more about the social capital construct, we formed an alternative social capital model (Figure 7.7) based on the interviews conducted within the company with seven *company selected* factors; (i) social connections, (ii) efficient usage of the social connections, (iii) social connections and career success, (iv) generalized norms, (v) togetherness, (vi) everyday sociability, (vii) volunteerism, ($X32, X33, X34, X36, X37, X38, X40$, respectively).

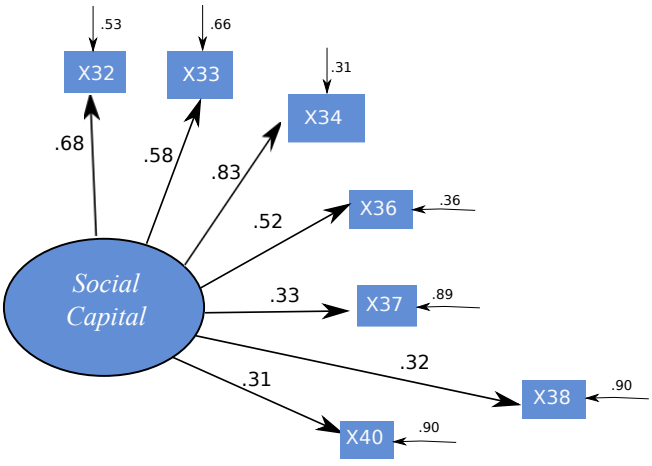


Figure 7.7: *Model VI Loadings with six factors of Social Capital based on Company Selected Parameters*

The second social capital model (Figure 7.7) had structural correlations that were statistically significant. All factor loadings were above the threshold of .20

where they ranged between .31 and .83. A null hypothesis was rejectable where χ^2 for independence model with 21 Degrees of Freedom is 334.324. The model showed a good fit with data, where $\chi^2(14, N = 192) = 21.311, p < .009$, and $RMSEA = .0523, GFI = .969, AGFI = .938, CFI = .973, NNFI = .960$). A χ^2 difference test indicated that there was a significant advancement between the independence model and the hypothesized model as $\Delta\chi^7(7, N = 192) = 313.013, (p < .001)$.

7.6 Models with Two Latent Constructs

In this part of our analysis, we built a series of models to investigate the correlation between productivity and social productivity, and social capital and social productivity, all of which were based on the indicators that were potentially affecting these latent constructs. To preserve the reliability of our SEM models, we used a limited set of indicators. These indicators, however, were previously discussed with the management team of Simurg, which was found important due to their past experiences. We used the data to test both the measurement and structural models.

Model VII (Figure 7.8) shows the relationship between productivity and social productivity with a set of factor loadings that were statistically significant. The factor loadings were between .21 and .69. The independence model, which tests the null hypothesis where all variables uncorrelated was clearly rejectable. The χ^2 for independence model with 66 degrees of freedom was 505.161. The proposed model yielded a good-fit, where $\chi^2(53, N = 192) = 88.125, p < .001$, and $RMSEA = .0589, GFI = .929, AGFI = .895, CFI = .914, NNFI = .893$). The significant improvement fit between the hypothesized and interdependence model was found by using a χ^2 difference test, $\Delta\chi^2(13, N = 192) = 417.036, p < .001$).

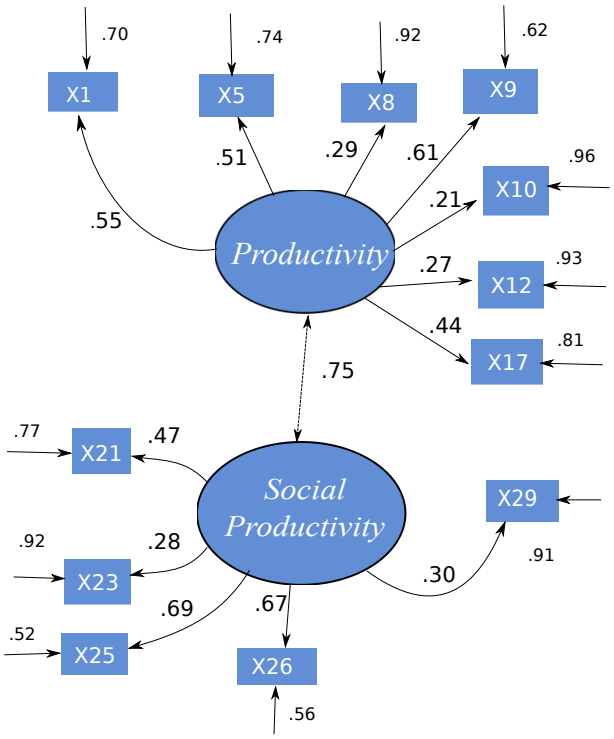


Figure 7.8: *Model VII for Productivity and Social Productivity in a Software Development Organization*

The most striking observation to emerge from the data comparison for the productivity construct was the work environment (*Standardized Path Coefficient*= .61). The motivation (*Standardized Path Coefficient*= .55) was the second significant predictor of productivity. For the social productivity construct, the collective team memory (*Standardized Path Coefficient*= .69), and unity of a team in the service of team goals (*Standardized Path Coefficient*= .67) were the two significant predictors. In addition, we observed high structural correlations between the latent variables (productivity and social productivity) $(.75, p < .05)$.

Model VIII has two latent variables, i.e. social productivity and social capital (Figure 7.9). The indicators were selected by the management team of the software company where all factor loadings yielded statistically significant values $(p < .05)$. The null hypothesis is not acceptable. The χ^2 for independence model with 36 Degrees of Freedom was 332.483. The significant improvement fit between the hypothesized and interdependence model was found by

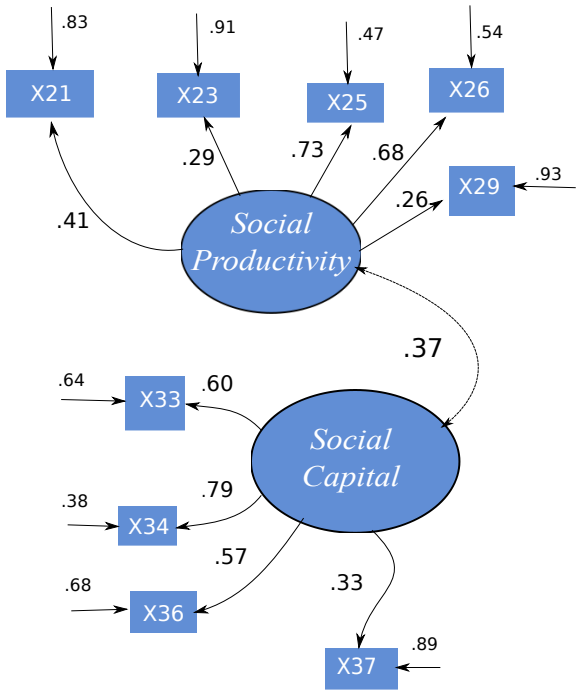


Figure 7.9: *Model VIII for Social Productivity and Social Capital in a Software Development Organization*

using a χ^2 difference test, $\Delta\chi^2(10, N = 192) = 278.805, p < .001$). The proposed model yielded a good-fit, where $\chi^2(26, N = 192) = 53.678, p < .001$, and $RMSEA = .074, GFI = .941, AGFI = .898, CFI = .903, NNFI = .87$).

From the data in Figure 7.9, it was apparent that for construct of the social productivity, predictors namely collective team memory (*Standardized Path Coefficient*= .73), and unity of a team in the service of team goals (*Standardized Path Coefficient*= .68) relatively got higher values. For the social capital construct, the less surprising predictors were the two indicators titled as efficient usage of the social connections (*Standardized Path Coefficient*= .60), and social connections for career success (*Standardized Path Coefficient*= .79). Moreover, correlation between social productivity and social capital was found with a statistically significant value of ($.37, p < .05$).

7.7 Refined Structural Equation Models

In the next part of our analysis, we combined related survey questions into categories by arranging them regarding their commonalities to reflect broader themes. Therefore, we formed new predictors that are summarized in Table 7.4. We calculated the average scores for these new themes in an attempt to improve our accuracy in measuring the latent constructs. For example, the average scores were calculated for factors of productivity related to team issues $X13$, $X14$, $X17$, all of which were combined to form a new category $Y7$. Concurrently, complexity issues were formed by the average scores from the factors classified as $X6$ through $X8$ to form $Y3$.

Table 7.4 presents all factors that were transformed to Y with calculated means, variances, and standard deviations as descriptive statistics. Once again, we calculated the reliability coefficients called the Cronbach's α for updated survey. It was found as .8, which confirmed that data was suitable for building models.

Factor ID	Factor Name	New Factor ID	mean	s.d.	var.
X1 - X2	Motivation	Y1	4.64	0.48	0.23
X3 - X4 - X5	Management Quality	Y2	4.00	0.59	0.35
X6 - X7- X8	Complexity Issues	Y3	3.71	0.61	0.37
X9	Work Environment	Y4	4.17	0.74	0.55
X10	Re-usability	Y5	3.80	0.97	0.93
X11 - X12	Requirements Stability	Y6	3.95	0.67	0.45
X13 - X14 - X17	Team Issues	Y7	3.52	0.48	0.23
X20 - X21	Team Leader	Y8	4.08	0.62	0.38
X22 - X23	Social Interaction and com.	Y9	3.98	0.67	0.45
X24 - X25	Information Awareness	Y10	4.14	0.60	0.36
X26 - X27	Team Cohesion	Y11	3.98	0.69	0.48
X28 - X29	Fairness	Y12	3.79	0.71	0.51
X30	Frequent Meetings	Y13	3.96	0.95	0.90
X31	Social Trust	Y14	4.29	0.67	0.45
X32 - X33	Neighborhood Connections	Y15	3.66	0.81	0.65
X34 - X35	Group Characteristics	Y16	3.40	0.80	0.65
X36	Generalized Norms	Y17	3.21	0.92	0.85
X37	Togetherness	Y18	2.21	1.05	1.10
X38	Everyday sociability	Y19	3.27	1.13	1.27
X39 - X40	Volunteerism	Y20	3.40	0.85	0.72
X41	Experience and Trust	Y21	3.84	0.81	0.66

Table 7.4: *Means, Variances and Standard Deviations of the Combined Factors*

In all three models below, we had seven factors to identify our three constructs, hypothesized to covary with each other. Model IX (Figure 7.10) displays the relationship between productivity and social productivity by using a set of

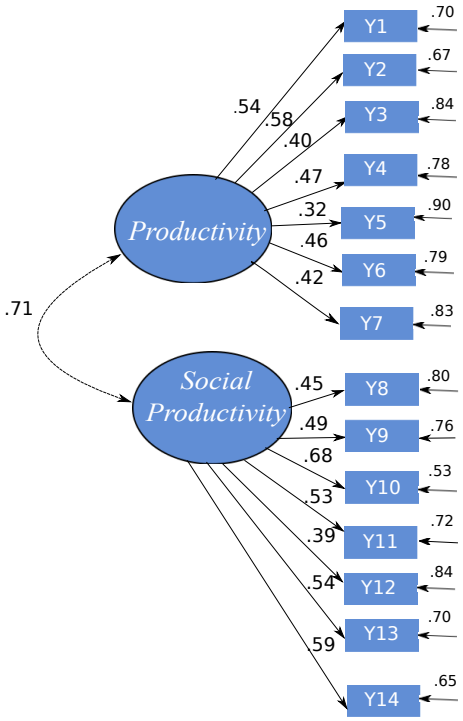


Figure 7.10: *Model IX for Productivity and Social Productivity in a Software Development Organization*

factor loadings that were statistically significant. The factor loadings were ranged between .32 and .68. The independence model was clearly rejectable. The χ^2 for independence model with 91 degrees of freedom was 858.748. The proposed model yielded a good-fit, where $\chi^2(76, N = 192) = 119.360, p < .001$, and $RMSEA = .0547, GFI = .92, AGFI = .89, CFI = .94, NNFI = .925$). To asses the improvement between the hypothesized model with the interdependence model a χ^2 difference test was conducted, $\Delta\chi^2(15, N = 192) = 739.388, p < .001$).

Empirical findings suggest that productivity was mostly defined by the factor called management quality (*Standardized Path Coefficient*= .58) and secondly by the factor called motivation (*Standardized Path Coefficient*= .54). From the social productivity viewpoint, the most significant factor was information awareness (*Standardized Path Coefficient*= .68) and the second important indicator was social trust (*Standardized Path Coefficient*= .59). The correlation between productivity and social productivity was .71.

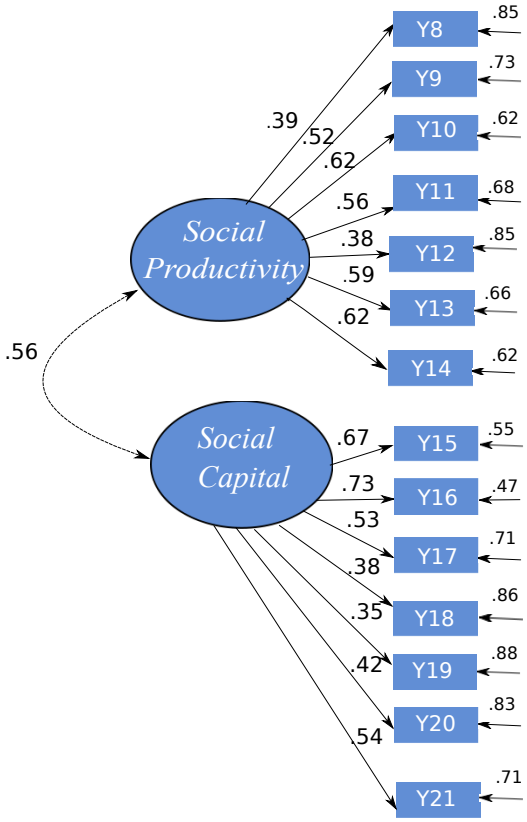


Figure 7.11: *Model X for Social Productivity and Social Capital in a Software Development Organization*

The model (Figure 7.11) depicts the relationship between social productivity and social capital by using a set of (statistically significant) factor loadings that ranged between .35 and .73. The null model was clearly rejectable. The χ^2 for independence model with 91 degrees of freedom was 967.046. The proposed model yielded a reasonable fit, where $\chi^2(76, N = 192) = 130.088, p < .001$, and $RMSEA = .0610, GFI = .911, AGFI = .88, CFI = .931, NNFI = .918$). To asses the improvement between hypothesized model and the interdependence model, a χ^2 difference test was conducted, $\Delta\chi^2(15, N = 192) = 836.958, p < .001$).

These results suggest that information awareness (*Standardized Path Coefficient*= .62), and social trust (*Standardized Path Coefficient*= .62) were the most important two factors of social productivity whereas neighborhood connections (*Standardized Path Coefficient*= .67), and group characteristics (*Standardized Path Coefficient*= .73) were the factors affecting social capital construct. Fur-

thermore, there was a significant correlation between social productivity and social capital, which was measured as .56.

7.8 The Tripartite SEM Model

Finally, we constructed a tripartite unified SEM model as Model XI (Figure 7.12) to investigate the relationships between productivity, social productivity, and social capital, and to show the factors affecting these latent constructs. To measure the hypothesized influence between the observed and latent variables, we built a model with three constructs, all of which were found statistically significant ($p < .05$) and ranged between .30 and .73. The independence model was clearly rejectable where the χ^2 for independence model with 210 degrees of freedom is 1680.137. The proposed model yielded a good-fit, where $\chi^2(186, N = 192) = 296.896, p < .001$, and the fit indices for the tripartite model were $RMSEA = .0559, GFI = .90, AGFI = .84, CFI = .914, NNFI = .90$). Furthermore, a χ^2 difference test was conducted, $\Delta\chi^2(24, N = 192) = 1383.241, p < .001$). Management quality (*Standardized Path Coefficient*= .59) was a significant predictor for productivity, which was followed by motivation (*Standardized Path Coefficient*= .53) and work environment (*Standardized Path Coefficient*= .47).

The most significant predictor for social productivity was found to be information awareness (*Standardized Path Coefficient*= .65), which was followed by the predictors of social trust (*Standardized Path Coefficient*= .60), and fairness (*Standardized Path Coefficient*= .56). The most significant predictor for social capital was group characteristics (*Standardized Path Coefficient*= .73), which was followed by neighborhood connections (*Standardized Path Coefficient*= .68). In addition, all of the structural correlations among the latent variables were statistically significant. The correlation between productivity and social productivity was .70; social productivity and social capital was .55, and productivity and social capital was .48.

Table 7.5 provides a summary of the important values of all the models used in the analysis.

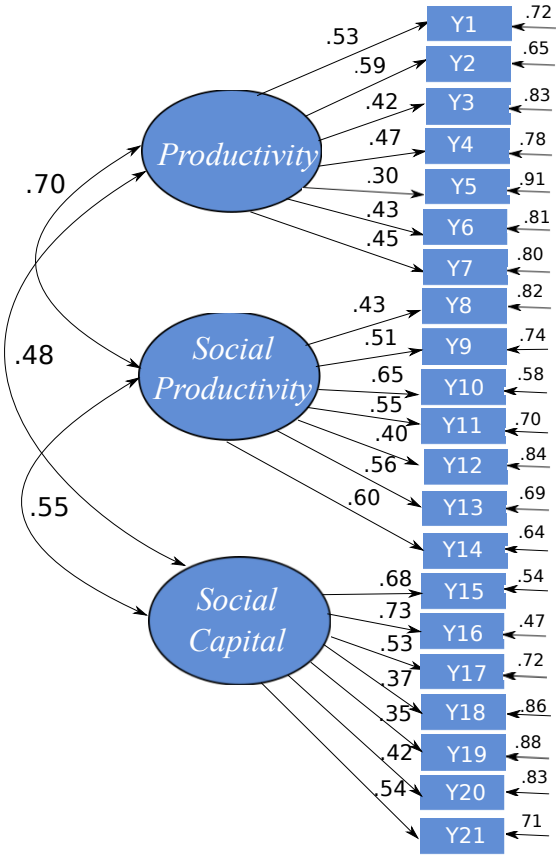


Figure 7.12: *Model XI for Productivity, Social Productivity and Social Capital in a Software Development Organization*

Model ID	χ^2/df	*RMSEA*	*GFI*	*AGFI*	*CFI*	*NNFI*	*Acceptance*
I	1.74	.0621	.887	.855	.823	.798	Yes
II	2.72	.095[1]	.89	.90	.85	.78	No
III	1.70	.0622	.964	.93	.934	.901	Yes
IV[3]	1.15	.0285	.969	.945	.99	.986	Yes
V	2.48	.088[2]	.951	.901	.913	.869	No
VI	1.52	.0523	.969	.938	.973	.96	Yes
VII	1.66	.0589	.929	.895	.914	.893	Yes
VIII	2.06	.074	.941	.898	.903	.87	Yes
IX	1.57	.0547	.92	.89	.94	.925	Yes
X	1.71	.061	.911	.88	.931	.918	Yes
XI	1.59	.0559	.90	.84	.914	.90	Yes

[1,2]Model was rejected because of its RMSEA value.
[3] Model was the most parsimonious of all the models tested.

Table 7.5: *Goodness-of-Fit indexes for all Constructed Structural Equation Models (refer to Table 3.5 for cut-offs)*

7.9 The Impact of Teams and Roles to Productivity, Social Productivity, and Social Capital

In this part of the analysis, we categorized the latent constructs with respect to the identified roles in Simurg. For each role identified by our survey, we calculated means, variances, and standard deviations, that is descriptive statistics, presented in the Table 7.6. To test the homogeneity of the data, here we calculated three coefficients of variation (CV), which is the percentage ratio (a comparison) of standard deviation to mean (see Equation 7.3). The data is called homogeneous when CV is below 33%, while values above cut-off value signify that there are outliers or some unwanted measurement errors that can affect the outputs. Since our coefficients fell within the threshold value, we confirmed that the data was homogeneous.

$$CV = \frac{s}{\bar{X}}(100) \tag{7.3}$$

Roles	Productivity	Social Productivity	Social Capital
IT Specialist	3.82	4.00	3.40
Project manager	3.85	4.01	3.23
Software architect	3.93	4.10	2.85
Software developer	3.88	3.97	3.29
Team Leader	3.89	4.04	3.26
Software Tester	3.96	4.29	3.60
Software specialist	3.96	4.00	3.39
System Analyst	3.70	3.82	3.20
System Engineer	3.64	3.82	3.70
Mean($\bar{X}$)	3.85	4.01	3.32
Standard deviation(s)	0.11	0.14	0.25
Coefficient of variation (%)	2.92	3.58	7.40

Table 7.6: *Roles versus the Means, Standard Deviations, and Coefficient of Variation for the Social Constructs*

To investigate the difference between the ideal and the actual team size for Simurg, we asked two questions in our survey (see Appendix A). Question 18; *How many members are in your immediate development team* ($TEAMSIZE$), and Question 19; *In your view, how many of your team members are operating at high levels of productivity*, ($IDEAL\_TEAM\_SIZE$). Using this information, we derived three variables namely, $EXCESS\_TEAM\_SIZE$, which was identi-

fied by actual team size minus ideal team size. $WHETHER\_IDEAL\_TEAM$, a boolean variable, which can be true or false (zero or one), and finally a variable called $UNDER\_IDEAL\_OVER$. In addition, we asked the participants about their years at the industry ($WYEAR$), and the years they spend in this company ($WTHISFIRM$). The descriptive statistics with the averages of constructs defined for the team-based variables with respect to the role of individuals were presented at Table 7.7.

Roles	WYEAR	WTHISFIRM	TEAMSIZE	IDEAL TEAMSIZE	EXCESS TEAMSIZE
IT Specialist	7.14	1.52	5.36	3.84	1.52
Project Manager	12.18	3.06	5.94	3.71	2.24
Software Architect	17.25	4.50	10.00	6.00	4.00
Software developer	5.67	3.14	7.44	4.80	2.64
Team Leader	10.77	2.85	8.38	4.85	3.54
Software Tester	3.85	2.00	7.30	5.04	2.26
Software Specialist	1.88	1.41	8.28	7.24	1.03
System Analyst	8.20	4.40	14.80	9.10	5.70
System Engineer	13.40	2.20	8.40	5.40	3.00
Mean	8.93	2.79	8.43	5.55	2.88
Standard deviation	4.91	1.13	2.76	1.71	1.40

Table 7.7: *Mean Scores of Roles versus Team Constructs*

This table is highly revealing in several ways. Firstly, it shows the average of years of experience both in this organization and outside, and as a whole, different roles identified by the survey. Secondly, it is apparent from this table that software architects have the highest experience average, and system analysts work in the biggest teams. By comparing with other roles, software specialists and IT specialists on the other hand think that they are working close to the ideal team size. From this data, we can see that the lowest value for years of experience both in this firm and in general are found as software specialists.
Furthermore, the results indicate that, of the 192 participants who completed this part of the questionnaire (see Appendix C), 80 participants (22 female, 58 male) thought that they were in a team that is in the ideal size, while 112 participants (25 female, 87 male) believe that their actual team is not at the ideal size. The role of participant with respect to their belief in under, ideal, and over-sized teams are shown in the Table 7.8.

Role	Under-sized	Ideal Team size	Over-sized
IT Specialist	0	13	12
Project Manager	0	5	12
Software Architect	0	1	3
Software developer	2	21	43
Team Leader	0	5	8
Software Tester	1	10	12
Software Specialist	0	19	10
System Analyst	0	4	6
System Engineer	0	2	3
Total Personnel	3 (%.02)	80 (%.42.98)	109 (%57)

Table 7.8: *Roles versus Participants Thoughts on Team Size*

From the data in Table 7.8, it is apparent that 57.02% of survey participants thought that their team was not in ideal size. What is interesting in this data was that many of the software developers think that they were in an over sized team. However, there were only two software developers and one software tester who thought that they might need additional members to their teams to reach the ideal team size.

Turning now to the experimental evidence based on our survey, we seek the degree of casual (strength of) relationships between different variable pairs. One way to investigate the linear relationships between a pair of variables is to construct a correlation structure. To understand how the data trends together, the relationship can be quantified by a coefficient called correlation coefficient. It is a coefficient that measures how strongly the variables are connected, and what values they take between -1.0 and $+1.0$. The minus sign shows the changes in the negative direction (i.e. inverse relationship), so when the correlation is $+1.0$, it is called a perfect positive correlation. Using a set of n observation of a pair of variables, $(X_1, Y_1), (X_2, Y_2), ...(X_n, Y_n)$, the correlation coefficient for this part of the study was calculated by following equation 7.4.

$$r = \frac{\sum_{i=1}^{n} (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n} (X_i - \bar{X})^2} \sqrt{\sum_{i=1}^{n} (Y_i - \bar{Y})^2}} \tag{7.4}$$

To calculate the significance of correlation (r), a T-test can be performed, and

calculated by equation 7.5. The test comprise of a comparison of a cataloged t value with respect to an empirical one.

$$|t| = \left| \frac{r}{\sqrt{1 - r^2}} \times \sqrt{n - 2} \right| \tag{7.5}$$

where n is the number of roles, and r is the correlations (n=9 in our case), n-2 is the number of degrees of freedom. Table 7.9 provides the significant correlations and values of T-tests, which were used to analyze the relationship between pairs of variables.

Variable Pairs	r	t[1]
WTHISFIRM - TEAMSIZE	.68	2.45
WTHISFIRM - Social Capital	-.76	-3.05
WTHISFIRM - EXCESS_TEAM_SIZE	.86	4.49
IDEAL_TEAMSIZE - TEAMSIZE	.91	5.75
TEAMSIZE - EXCESS_TEAM_SIZE	.86	4.48

[1] $t_{critical}$ (df=7, 0,05)=2,3651, $p < .05$

Table 7.9: *Statistically Significant Pairwise Correlations for Roles from the Survey*

Table 7.9 illustrates that the years participants spent at the company and the team size had a positive correlation, .68, and the correlation between the years they spent at the company and excess team size was .86, whereas a strong negative correlation, $-.76$, was observed between the years participants spent at the company and the value that participants gave to social capital. We confirm that the longer period participants worked in the company, the bigger teams they started to work with where they tended to think that their team was too large to obtain higher productivity.

Interestingly, participants who spent more time with Simurg were inclined to give less importance to social capital (see Table 7.9). Furthermore, the correlation between team size-ideal team size was positive and higher than the correlation between team size-excess team size. We conclude that for those participants who work in a larger team size, their ideal team size gets higher, and they also tend to think that their team size was not ideal for software development productivity.

Recently, there have been several empirical investigations into the effects of team

size on software development productivity [316]. However, since Brooks [317] initiated a discussion about the possible effects of team size on the productivity of software development, team size has become a central issue for empirical research in software engineering. From a socio-technical perspective, the System Dynamics model was developed to investigate the human capabilities such as planning a set of possible staffing procedures on a variety of project costs with different schedules [318]. In addition, studies of software development productivity showed the importance of the average team size [241]. Most importantly, the findings of the current study were consistent with those of Putman [319], who found evidence that the productivity of software development was found higher for smaller software teams. Recent evidence from a number of management studies suggests that small teams are performing better [320]. In particular, a study indicated that size of the most effective software teams varies between 3 to 6 members [321]. Taken together, our findings further support the recent investigations in team size for software development projects.

7.10 Case Study I: Threats to Validity

Here, we consider several potential threats that were addressed for the validity of case study I. To deal with construct validity issues, first we conducted a number of literature reviews to build our theoretical model; secondly we asked a group of experts from both academia and industry to assess our initial constructs and the potential factors that are the representatives of the constructs being measured. It was suggested to conduct an initial implementation in an industrial focus group in order to check the validity of our research questions and conduct a test study with our preliminary ideas. Then, we published the initial results of a pilot study and got some early feedback before conducting the industrial case study. In brief, our initial research questions were taken from a purely theoretical perspective and aligned with practical industrial viewpoint. In addition, we revised our survey questions based on the initial comments from experts to increase the clarity of items. After conducting the survey, for the first part of the study, we used Cronbach's α to test the reliability of our

identified constructs. For the second part, we checked the data homogeneity by using the coefficients of variation. In accordance with these, we believe that both latent and observed variables, and pairwise correlations in this study have been measured properly.

To cope with internal validity problems, first we built a number of models based on the selected constructs and tested them with a set of factors identified from the literature and refined through focus groups. Secondly, we selected the most convenient time for participants to start the survey (we had to wait for a while to capture such a time frame), and limited the time for respondents to two weeks time to avoid any history effect. Thirdly, the measures taken through the survey were collected consistently (i.e. without changing the dependent variables in the survey instrument) so as to deal with any instrumentation effect. Fourthly, participants could give biased responses, as they may not behave rationally, therefore, once again, we checked the internal reliability and validity of each question using the Cronbach's α calculations.

To manage external validity problems, we iteratively built a number of SEM models similar to a conceptual replication tests (i.e. an alternative perspective that test the same concept in different ways) [121]; we tested our three constructs with different factors and investigated their correlations by building different combinations. As a part of case study I, we conducted validation interviews in which the measured factors were reviewed by a group of experts who had already contributed to the several aspects of the study; (i) to check the validity of the identified factors and their importance, (ii) to check whether the findings are generalizable. Finally for the reliability aspect, we clarified the data collection method and documented the processes and the protocols that were developed.

7.11 Chapter Summary

This chapter detailed the data analysis of Case Study I. The theoretical models of productivity, social productivity, and social capital of software development are derived from literature and presented in Chapter 5. They were empirically investigated using techniques such as structural equation modeling, and correlation analysis. In the last part of this chapter, the impact of teams and roles were investigated with respect to the productivity, social productivity, and social capital of software development organizations. Overall, we confirmed that there is a significant correlation between measured social productivity and the productivity of software development organization. Furthermore, our empirical results indicated a strong negative association between the value of social capital and the time individuals had spent with the software development company. Overall, these findings suggest an important role of the social aspects of software development in software development productivity. The next chapter will discuss the results from Case Study II.

Chapter 8

Empirical Findings: Case Study II

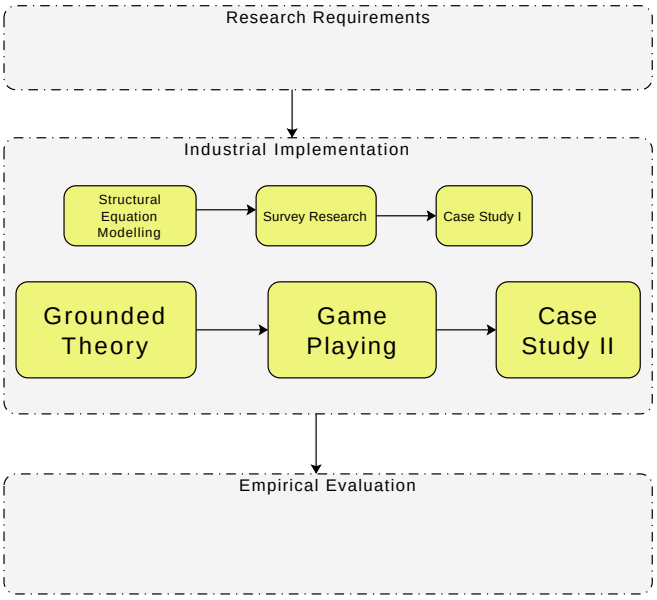


Figure 8.1: *A part of the conceptual overview of the research*

8.1 Introduction

This chapter presents the major empirical findings of the second case study that demonstrates an approach, which has the potential to help software development managers to relate personality types of practitioners with team structures for better team configurations. To this end, we develop a card game for measur-

ing personality traits compatible with the MBTI[1], which is the most frequently used tool in the industry for profiling software development personnel [280]. Furthermore, we demonstrate a technique to visualize effective team structures using social characteristics of their personnel for exploring team compatibility. Similar to the psychological evaluation tests our technique is based on situational context cards that are accompanied by MBTI-like questions, which are derived from several situations captured from the events observed in the software industry. Grounded on software development concepts, these context cards are used to create a game-based approach where the goal of the game is to reveal personality types of software practitioners. The method for the creation of situational context cards, rules and the structure of a game are detailed in this chapter. After validating the instrument, an industrial implementation is conducted and lastly results found by the case study are illustrated as an MBTI-Team radar (i.e. a spiderweb chart) using five software development teams from Simurg.

8.2 Crafting the Instrument and Protocols

This part of the study is comprised of several sequential steps, which requires a significant amount of time on part of participants. Therefore, the initial process of card creation was simultaneously conducted both at a university environment and in an industrial setting.

Furthermore, we intentionally used Keirsey's temperament sorter template as a reference framework for our assessment for two reasons. First, it is the only version of the MBTI test on the market freely available (via Keirsey's book). We thought using a common template would be helpful for evaluators to work on our cards. Secondly, we preferred to be compatible with MBTI tests for the outputs of this study. Our approach comprises three main steps: (i) initiation phase, (ii) card creation phase, (iii) comparison phase (see Figure 8.2).

[1]Personality research literature in software engineering are reviewed in Chapter 6.

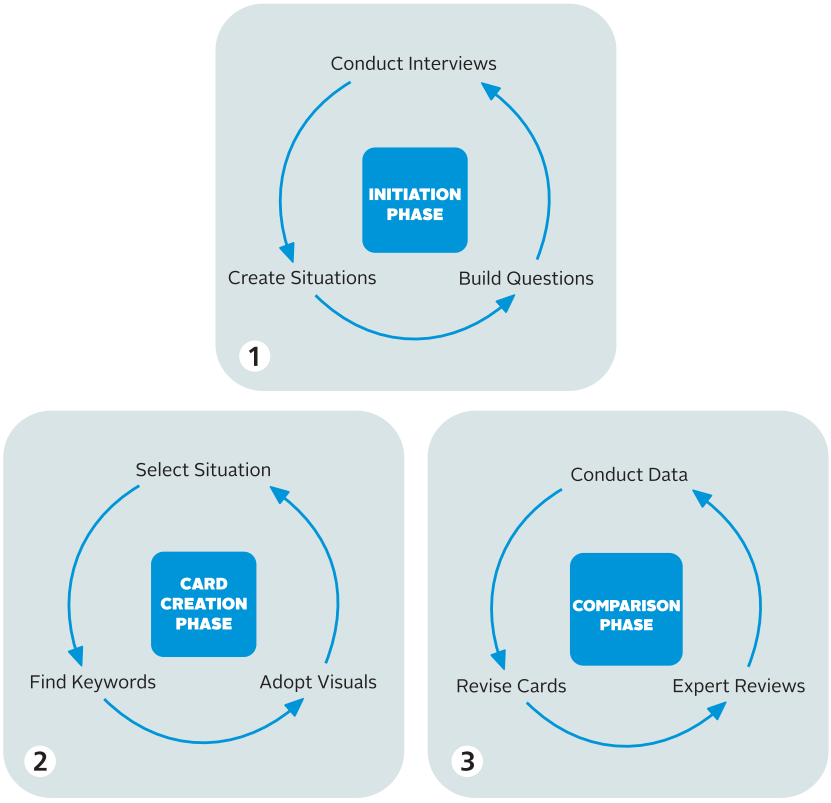


Figure 8.2: *The Systematic Process for Creating Context Cards*

8.2.1 Initiation phase

The initiation phase started with searching for a set of context dependent situations from the industrial settings that are transformable into hypothetical situation, which were later used in the card design process. To improve the findings through peer confirmation, we reviewed the questions of Keirsey's Sorter [281] for potential themes to which our context dependent questions should be based upon. To store the codings transcribed from the interviews, a codebook was also created (see Figure 8.3).

8.2.1.1 Initial Interviews

By following Creswell's advice [143], we selected 20 participants - who were highly experienced in software development - for this part of the study (10 were selected for semi-structured interviews, and 10 were selected for the ex-

pert reviews). From among the initial cohort of 60 software practitioners who are willing to participate in the study, we selected 10 individuals for the interviews based on their availability, work experience and age. All participants were above 30 years of age, and they had at least five years of experience in software development domain. Designated interviewees were informed about personality type research in software engineering and requested to submit a conventional MBTI based test by email. Next, they received a consent letter and a set of interview questions such as "*What do you think is missing in a personality test like this for identifying software engineers personality types?*", "*Can you think of any domain specific situation that can replace a question from the questionnaire?*", etc. Later, follow-up discussions about the MBTI types were recorded for transcription. During the process, the transcriptions were segmented based on several coded parameters such as similarities and contradictions in speech, and then they were processed before starting the analysis (see Appendix B for sample coding).

Based on the preliminary outcomes, 5 people out of 10 interviewees were selected for half an hour one-to-one interviews. For the next iteration, we selected 3 participants who were from the research and development department of the company for a one hour extra discussion. Finally, we discussed the findings of the previous iterations with the head of the human resources of Simurg.

Furthermore, additional data concerning to a number of business situations were collected by one of the researchers who participated in a number of meetings such as interviews and focus group studies.[2]

Participant ID	Title	Age	Years of Experience	Education
P12	IT Specialist	33	6	MSc.
P36	Project Manager	47	7	PhD.
P44	Software Architect	37	12	BSc.
P57	Software Developer	31	6	BSc.
P99	Software Developer	33	7	BSc.
P106	R&D Team Lead	39	14	PhD..
P112	Software Tester	32	4	MA.
P73	System Analysis	34	9	BA.
P51	R&D Team Member	32	7	MSc.
P97	R&D Team Member	31	5	MSc.

Table 8.1: *Participants' Information*

[2]The interview data are kept confidential.

Table 8.1 outlines the profile of the 10 participants including their roles (titles), age, years of experience and level of education. The first iteration was conducted on ten people. The interviews were analyzed from audio recordings by using a tool for scientific transcription called *f5*. Additionally, we used the TAMS analyzer as a Computer Assisted Qualitative Data AnalysiS (CAQDAS) tool while processing our data.

8.2.1.2 Validation of the Codebook

To aid in coding the segments of information, we used a set of *mnemonic codes* to designate the roles of software practitioners (e.g. SD, ST, etc.), which were formed in Chapter 6. The goal was to create an index between roles, themes and keywords, which might be beneficial for the categorization process. The data segment, which had a potential keyword was given a new code and added to a codebook. When the transcription was completed, the researcher reviewed each independent information segment for the themes of interests where these segments were checked for potentially new concepts. Whenever researchers found a potential item stored in a transcribed segment, they compared it with the initially coded themes and constructs.

To validate our codebook, the coding scheme was discussed with an individual from the research and development team of the company who has both academic and industrial experience. Initial results allowed us to derive the initial personality type keywords by open coding, and further we enhanced our results by using several documents and reports where the themes were for (E/I); (i) social interactions, (ii) social courage, and (iii) being conversationalist. For (S/N), (i) factuality and fiction, (ii) experience and hunches, (iii) specifications and generalizations. For (T/F); (i) being firm or gentle, (ii) personal values and generalized principles, (iii) thoughts and evidence versus feelings. Lastly for (J/P); (i) plan ahead or adapt as you go, (ii) product versus process, (iii) Act quick and decide fast were found as the themes. In addition, the TAMS analyzer allowed us to tag a set of potentially suitable keywords for the next phase. To select the keywords for our deck of 70 cards, we analyzed the frequency of

similar keywords with respect to different interviewees. These keywords were discussed with several rounds of feedback and revisited by a user experience designer who also guided the picture selection process. The mapping between the identified categories versus constructed question cards are presented in Table 8.2.

Identified Themes	**Derived Questions on Cards**
Social interactions	1, 8, 15, 22
Social courage	29, 36, 64
Skills and confidence in conversations	43, 50, 57
Being factual or fictional	2, 17, 23, 24, 30, 44
Relying on experiences or hunches	9, 10, 16, 37, 45, 51
Focusing on specification or generalizations	3, 31, 38, 52, 58, 59, 65, 66
Being firm or gentle	4, 5, 19, 25, 32, 46, 53
Focusing on values or principles	18, 26, 47, 60, 67, 68
Valuing thoughts or feelings	11, 12, 33, 39, 40, 54, 61
Planing a head or adapt as you go	6, 13, 20, 27, 35, 42, 56, 63
Valuing process over product	14, 21, 41, 69, 70
Acting quick and deciding fast	7, 28, 34, 48, 49, 55, 62

Table 8.2: *The identified themes with respect to the derived questionson Cards*

In an effort to validate the keywords extracted from preliminary resources such as Keirseys work and with the selected situations, *a concept map* [97] was created particularly based on the themes and the keywords (see Figure 8.3). After conducting a theme analysis of the transcripts from the previous step, the codebook was validated using the information stored, and the frequency of each keyword in transcript was categorized for further analysis. The examination and comparison of the data being as a continuous process where findings were taken back to selected participants from the study until an expected saturation has been either confirmed or verified, whenever a situation was identified, it was compared with several other previous situations for understanding their similarities as well as the differences. After finishing these tasks, we used axial coding to organize and combine the keywords with the selected categories from the transcribed documents and memos. Figure 8.3 represents a part of the codebook based on the extracted keywords and personality traits.

8.2.2 Card Creation Phase

We prepared the initial version of the cards by using both the business situations and Keirsey's temperament sorter questions that we defined during the first

phase. For each of the 70 cards (see Appendix I), we selected a hypothetical situation and a keyword from the codebook. Next, we chose a picture with creative commons license, which reflected the keyword and the situation that was previously formed (see Figure 8.4 for a sample card).

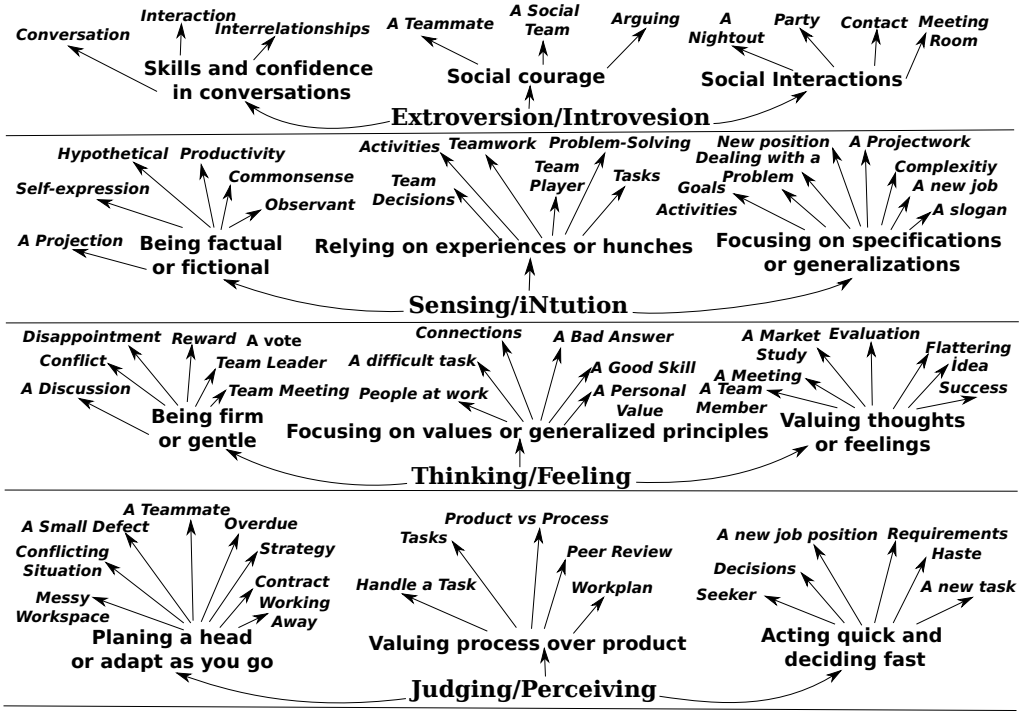


Figure 8.3: *Illustration of the codebook extracted from emergent keywords*

8.2.3 Comparison Phase

Finally, in the third phase, the cards were revised based on both the data collected from the the next wave of interviews and the opinions of the experts from the industry. To substantiate the reliability of this part of the study, we consulted 10 experts (see Table 8.3) both from the software industry and academia to discuss our findings, and the hypothetical situations were evaluated. As a result, we designed 70 cards. All cards had two faces. The front had a picture and a keyword that defined the theme of a picture. The goal was to visually prepare the participants for the hypothetical situation that was written on the other side of cards. Each situation had two different answers, which indicated

the participants inclination on a trait, e.g. being introverted or extroverted.

Expert ID	Title	Age	Years of Experience	Education
E1	Software Manager	46	20	PhD.
E2	UX Designer	36	7	MSc.
E3	Graphical Designer	30	4	BA.
E4	Software Practitioner	31	6	BSc.
E5	Clinical Psychologist	43	16	PhD.
E6	Organizational Psychologist	39	11	PhD.
E7	Instructional Designer	38	9	MA.
E8	Academic	40	14	PhD.
E9	Academic	45	17	PhD.
E10	Academic	58	25	PhD.

Table 8.3: *Expert Reviewers' Information*

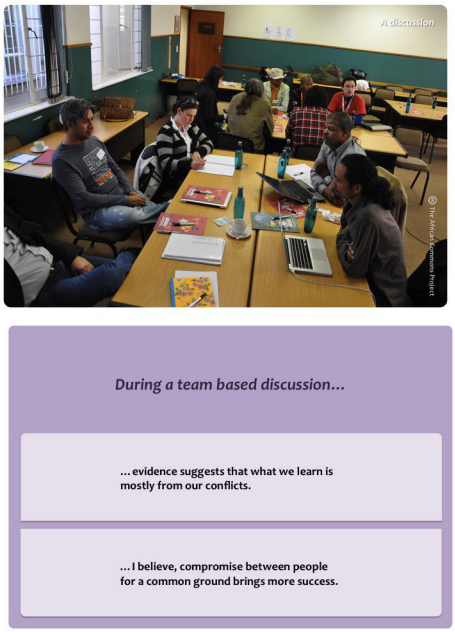


Figure 8.4: *A Two-faced Situational Context Card Example*

To sum up, the situation context cards (see Figure 8.4 for a sample) were designed to highlight situations mostly specific to software development domain where participants had always two answer choices embedded in each card (see Appendix I for all cards). A number of questions defined a cause and effect relationship and sought for a single answer or an action. The card game relied on the assumption that individuals would have increased tendencies to respond correctly while playing a game rather than answering a series of static questions. Ultimately, our goal was to introduce these cards to an individual or a group

of participants in a game form to facilitate the identification process of their personalty traits. Lastly, the processes of construction of situational context cards lasted more than 70 hours of work with Simurg.

8.3 Rules of the Game

Here, we define a game form titled *a game of revealing personality types*. Ultimately, the expected outcome for this game is finding the personality types of individuals. The actual players are a game master (GM) and members of a software team[3]. The winning condition for each player is to learn his or her true personality type by the help of our card based trait identification game.
A GM, as an administrator, has the main function to interact with the participants, thus operate the game. For example, a GM shall be showing the pictures of cards and read questions to them. The players or participants should follow instructions of GM. The game can be played by the members of the team together in 20-30 minute sessions. It shall be started by GM, who draws a card from the deck and shows the first card's picture to participants. He then starts reading the situation, which is written on the back side of the card. The cards also include two different directions with respect to situations defined on the cards. Participants use a preconstructed sheet to fill in their answers marking either a or b. For a one-to-one game, the game master might like to sort the cards based on their colors on the game table and later calculate the results of the experiment by counting the sorted cards. However, for multiple players, the template mentioned above shall be used to record the findings. There are 70 cards and questions, which shall be asked to participants. This experiment should also be conducted in a silent room, which should be performed without a break so as to preserve participants' concentration. Furthermore, the administrator (GM) should wait for multiple participants to mark their answers to interview form before starting the next question.

[3]Game Master could be a researcher, a practitioner or a software manager who has the basic knowledge to operate the game

8.4 Quantitative Evaluation of the Survey Instrument

8.4.1 Pilot Study I

To estimate the response reliability of our game cards, we conducted multiple measurements based on the selected 15 participants at a university environment (see Appendix E). In this part of the work, we used individuals, who were novice developers with at least a year of industrial experience. These individuals were picked by the criterion of either whether either they have worked together as a team for some projects or they are the individuals who worked in the same environment at least for sometime. All the sessions started with an introductory statement. We provided a response form for individuals so that they can mark their responses on the interview document. The interview form also had additional feedback questions that might be useful for the next iteration.

8.4.2 Pilot Study II

The reinterview survey was one of the most commonly used methods for the investigation the measurement errors [123]. According to Presser et al. "*...the reinterview survey was designed to replicate the original interview independently so that the measurements from the two surveys can be assumed to be parallel. By parallel measurements we mean measurements that have the same probability of false positive and false negative errors and whose errors are independent*" [124, pp. 229].

In our case, the respondents were recontacted six weeks after an initial pilot study and requested to participate in the same game-based card test once again. The primary goal here was to replicate the original process and to gather the required information from the same set of participants. The test/retest card game revealed the reliability and acceptability of responses where we asked the same questions once again within the same environmental conditions to the same participants and hence to identify the flawed questions [124]. However, variable errors could be observed in both interview (Pilot I) and reinterview (Pilot II) process. "*In this way, the measurement error variance associated*

with the original survey response could be estimated by the variation between the original and reinterview responses " [123, pp. 298].

8.4.3 Measuring the Reliability of Questions on Cards

To measure the reliability of cards, we used a common re-measurement method to construct and analyze the answers of participants for the same question in both pilot I and pilot II card tests by calculating the term called *index of inconsistency* (I) [322], where I would be considered as the ratio of question-level measure of response variance to the total response variances for a given question. The variable called the reliability ratio is also represented by a $1-I = \kappa$, where κ is called Cohen's measurement of reliability [323]. The index of inconsistency (I) can be represented as

$$I = \frac{g}{p_1 q_2 + p_2 q_1} \tag{8.1}$$

where $g = (b + c)/n$ is the disagreement rate, and the total sample size is denoted by $n = a+b+c+d$, where a is the number of participants who selected the first option in both runs, d is the number of participants who chose the second option in both runs, b is number of participants who choose the second option on the first run, and the first option on the second run, c is the number of participants who chose the first option on the first run, and second option on the second run. The ratio shows the answer *yes* in the original interview as $p_1 = (a + c)/n$. The *yes* answers in the reinterview is shown as $p_2 = (a + b)/n$. The value q_t = 1-p_t, for t = 1, 2, designates the proportion of the answer *no* for the interview and reinterview [124].

8.4.4 A Sample Calculation

To illustrate our basic idea stated above, we chose Question 18 (see figure 8.5), which can be considered as one of the questions that achieved a high level of consistency from the respondents. The question states: "When you are choosing the people you work with...". The dichotomous answers are as follows:

(a) "...the consistency and stability of their thoughts is the most important factor". And (b) "...harmonious relationships is the most important".

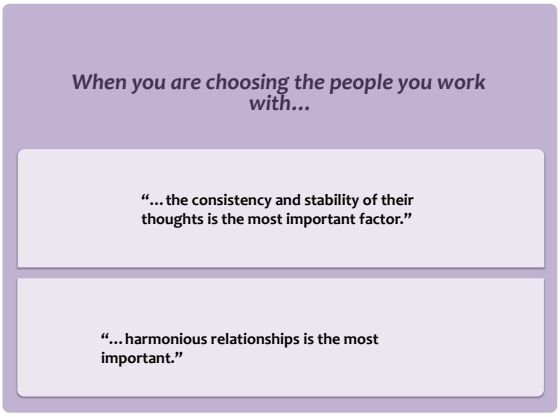


Figure 8.5: *Question 18 Selected from SCC as a Concrete Example*

The survey results for question 18 are depicted in Table 8.4 that have been labeled a, b, c, and d. As the retesting replicates the primary test, all measurements can be considered as identical.

	Interview Response	
Reinterview Response	*First Option*	*Second Option*
First Option	a=10	b=0
Second Option	c=1	d=4

Table 8.4: *Interview Reinterview Table for Question 18*

Table 8.4 is highly revealing in several ways. First, it shows that there are 10 persons who selected the first option in both runs. We observed only 4 participants who chose the second option in both runs. As Table 8.4 shows, there are no participants who chose the second answers in first run and chose the first option on the second run. In addition, we found only one participant who chose the first option in the first test and selected the otherwise in the second run. The index of inconsistency for the data in Table 8.4 is calculated as $I = 15.7\%$, therefore a high coefficient of reliability of $\kappa = 84.3\%$ can be measured.

According to Biemer et al. [123], the acceptability for the response consistency and reliability should be judged within these ranges:

$$Acceptability = \begin{cases} Good & I \leq .20 \text{ or } \kappa \geq .80 \\ Fair & .20 \leq I \leq .50 \text{ or } .50 \leq \kappa \leq .80 \\ Poor & I \geq .50 \text{ or } \kappa \leq .50 \end{cases} \tag{8.2}$$

In light of these remarks, we calculated all κ values for all the questions in the questionnaire[4]. Table 8.5 summarizes the number of questions found in predefined κ ranges. We selected 30% as the range for the cutoff values, and we found the questions; Q4, Q21, Q22, Q24, Q26, Q27, and Q31 below the expected reliability coefficient κ.

κ % Range	Number of Questions
0 - .30	7
.31 - .45	9
.46 - .60	10
.61 - .75	14
.76 - .90	30

Table 8.5: *The Range of κ numbers found for the entire survey instrument*

From this data, we can see that this part of the study yielded statistically significant results where only 7 of 70 questions were found problematic (one question from (E/I) trait, and two questions from each (S/N), (T/F), (J/P) traits were out of range). Therefore, we performed our calculations by dropping these questions, and the ultimate results are shown in Table 8.6.

What is interesting in this analysis is that extroversion was observed as a dominant dichotomy during the pilot study, which was somehow compatible with the recent findings in MBTI research in the field of software engineering (e.g. [298]).

MBTI Type	Number of Participants	% in Sample Population
ENFJ	2	13
ENTJ	2	13
ESFJ	3	20
ESFP	3	20
ESTP	1	7
INTP	1	7
ISFJ	2	13
ISFP	1	7
Total	15	100

Table 8.6: *Personality Traits found by Situational Context Cards in Pilot Study*

[4]The calculations for all questions can be found in the Appendix F.

8.5 Quantification of the Instrument: Average of Weights

Based on the hypothesized model constructed from the grounded theory perspective, this part of the work details these qualitative notions into a rigorous quantitative model of personality traits particularly form a novel approach to MBTI typology. To build a scale of measurement with quantitative characteristics from MBTI, we conducted a survey based on the categories identified in the previous section. In a classical viewpoint, Myers and McCaulley [276] suggest that MBTI is a qualitative sorter where each question has equal importance. However, unlike a typical MBTI test, we hypothesized that the significance of the questions should differ in terms of their impact on the final output and therefore should influence the results accordingly.

Further, this argument sets the stage for the quantification with respect to several subcategories which may eventually be combined in a scale to form a better instrument. In particular, it should have the ability for more precise measurement of personality traits. This also accorded with our previously conducted interviews, which reveals the fact that questions should be interpreted within a level of importance; therefore, they should not be considered equally.

In light of our grounded theory model, we divided each dichotomy in three sets of themes, which were identified by some keywords. By following our model and the saturated categories explained previously, we formulated that extroversion/introversion dichotomy can be identified by the indicators; (i) social interactions, (ii) social courage, and (iii) individual's skill and confidence in conversations. For sensing/intuition dichotomy, three factors proved more effective on identifying that trait; (i) being factual or fictional, (ii) relying on experiences or hunches, and (iii) focusing on specifications or generalizations (detail orientation). Thinking/feeling dichotomy was defined by (i) being firm or gentle, (ii) focusing on personalized values or generalized principles, and (iii) valuing thoughts or feelings. Finally, for judging/perceiving individuals preferences over (i) planing a head or adapt as you go, (ii) valuing process over product, and (iii) acting quick and deciding fast or otherwise.

Table 8.7 summarizes the indicators that are potentially affecting the person-

ality traits, the measurement of means, variances, standard deviations, and average of weights with respect to the dichotomies adapted from Myers-Briggs.

Factor ID	Description	Mean	S.D.	Var.	A. W.	Traits
P1	Social interactions	3.77	0.69	0.47	0.80	E/I
P2	Social courage	3.14	0.56	0.32	0.61	E/I
P3	Skills and confidence in conversations	3.50	0.79	0.63	0.74	E/I
P4	Being factual or fictional	2.96	0.70	0.49	0.65	S/N
P5	Relying on experiences or hunches	3.56	0.77	0.59	0.79	S/N
P6	Focusing on specification or generalizations	2.95	0.76	0.58	0.55	S/N
P7	Being firm or gentle	3.68	0.65	0.42	0.69	T/F
P8	Focusing on values or principles	3.64	0.69	0.48	0.75	T/F
P9	Valuing thoughts or feelings	3.06	0.55	0.31	0.51	T/F
P10	Planing a head or adapt as you go	3.33	0.57	0.33	0.68	J/P
P11	Valuing process over product	3.69	0.58	0.33	0.77	J/P
P12	Acting quick and deciding fast	2.87	0.78	0.61	0.60	J/P

Table 8.7: *Factors of Personality Traits, Descriptions, Means, Standard Deviations, Variances, Average of Weights, and Traits*

Using the final part of the survey instrument from the first case study (see Appendix A), we conducted a personality type survey (on a 4 Point-Likert Scale) to identify the importance of the factors that are affecting personality constructs. In light of the collected data, the impact of the questions to the personality assessment results have been calibrated based on the weights of each question.

In the quantification of the questionnaire, it was found that the responses had a high internal consistency (or reliability) where the overall questionnaire has a Cronbach α of .71. The weights of each question was on a four 4-point Likert scale ranging from *very important* (4) was assigned to a weight of 1, *important* (3) with a weight of .75, *moderately important* (2) with a weight of .50, and *of little importance* (1) with a weight of .25. The respondents were asked to indicate weights for each of the factors extracted from the schematic representation of grounded theory process (see Appendix E).

The average of weights can be calculated as follows:

$$\overline{\mathcal{W}_i} = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} w_{ij}}{n} \tag{8.3}$$

where m is the number of questions ($i = 1, ..., m$), n is the number of respondents ($j = 1, ..., n$), and $\overline{\mathcal{W}_i}$ is the average of weights for i^{th} question.

8.6 An Industrial Implementation

From an industrial point of view, a card game was conducted on-site on 63 software practitioners so as to determine their MBTI based personality profiles (see Appendix H for collected data). The participants were selected from a group of individuals from case study I. As the arrangements were kindly requested by the management, all individuals participated[5].

To quantify our test results based on our card game, first we calculated the factor correlations among the indicators affecting personality traits as shown in the previous section. To assign a weight for each question, we used the relationships between the SCC cards and the indicators found by the grounded theory, which are depicted in Table 8.7. While each question was identified by a category as shown in Table 8.2, the weighted factors for these categories were calculated by using the values found in survey analysis. To find the personality traits for each subject (e.g. being either E or I) , we counted responses, calculated the weighted points for each category and further compared the two weighted points.

The maximum value that extroversion/introversion trait can take was $(E/I)_{max} = 6.45$[6]. The results of this part of the study demonstrate that we were not only revealing the MBTI types of the participants but also calculating a level of extroversion/introversion value that may be comparable with other participants during an analysis. The other three traits of personality had the maximum values identified as follows: $(S/N)_{max} = 11.84, (T/F)_{max} = 11.46, (J/P)_{max} = 12.04$. Table 8.8 shows the number of industrial participants with their MBTI types and their percentage value in our sample population.

Of the study sample, we conducted our game based personality test on 7 Project managers, 24 Software developers, 8 Software Testers, 20 Software Specialist, and 4 System Analysts. The present study enhanced our understanding of the quantification of MBTI types. Instead of labeling a participant with a single trait (e.g. E or I), our empirical findings were based on the percentage of

[5]See software teams subsection for selection criteria.

[6]If we treat each questions with the same weight this value would be 9.

MBTI Type	Number of Participants	% in Sample Population
ENFJ	7	11.1
ENFP	9	14.2
ENTJ	8	12.7
ENTP	4	6.3
ESFJ	5	7.9
ESFP	3	4.8
ESTJ	1	1.6
ESTP	2	3.2
INFJ	3	4.8
INFP	8	12.7
INTJ	1	1.6
INTP	3	4.8
ISFJ	2	3.2
ISFP	2	3.2
ISTJ	5	7.9
Total	63	100

Table 8.8: *Personality Traits found by Situational Context Cards in an Industrial Setting*

traits. It provided a new understanding of MBTI dichotomies on a quantitative scale, which aimed to improve the significance level of MBTI scale especially for team-based evaluations. An implication of this is the possibility of having more precise measurements. For example, it is now possible to distinguish between an individual who is 90% extroverted and someone who is 51% extroverted. Based on the quantification scheme, the total percentages of the identified traits with respect to job titles of the sample population are shown in Table 8.9.

Title	**Quantity**	**Extroversion %**	**Sensing %**	**Thinking %**	**Judging %**
Project Manager	7	62	48	37	47
Software Developer	24	58	43	60	38
Software Testers	8	72	38	49	54
Software Specialist	20	57	40	42	46
System Analyst	4	62	45	39	47

Table 8.9: *Overall Average Percentage of the participants with roles versus their traits*

The results, as shown in Table 8.9, indicate that software testers were more extroverted than the other roles in the sample population followed by project managers and software analyst. Of the initial cohort, project managers were found more sensing type while software developers are the most thinking type. The majority of participants who responded to judging/perceiving trait felt that they were more inclined to be perceivers. This was also consistent with other personality traits studies (on non-software engineering domain), which mostly found judging/perceiving are almost equal in numbers amongst the general population.

8.6.1 MBTI-Team Radar

A radar chart (graph) is a visual method of illustrating multivariate data in a two dimensional polar chart for the analysis of multiple variables where a set of variables represented together, which is suitable as a tool for comparison among a set of items [324]. A common approach to interpreting a radar graph is to read the values plotted using the data points. According to Harris: "*Other than readability, there is no limitation as to the number of variables that can be included in a single graph. Whatever number of variables there are, they are distributed equally around the 360' of the circle. Each axis typically has a scale along which one characteristic element is plotted for each data series involved in the comparison. After all of the data elements have been plotted, adjacent data points in the same data series are generally connected by straight lines forming closed polygons.*" [325, pp. 320].

Although a variant of team radar graph has been used in software process improvement domain as "the agile team radar" (e.g. [326,327]), to the best of our knowledge there is no prior study that combines MBTI and Team Radar concepts.

This research proposes a novel use of MBTI-Team radar (or an MBTI radar chart) with (dichotomous) polar coordinates of four traits guide researchers to explore a team to visualize which traits (i.e. (E/I), (S/N), (T/F), and (J/P)) are dominating the group characteristics. To illustrate dichotomies, our radar form includes all eight personality types, each divided into two axis. We used a percentage scale (ranging between 0% and 100% and divided into four equal parts) inside a team radar, which guided us to represent an individual on a continuous and dichotomous scale.

Figure 8.6 illustrates the conceptual figure for an empty MBTI-Team radar graph.

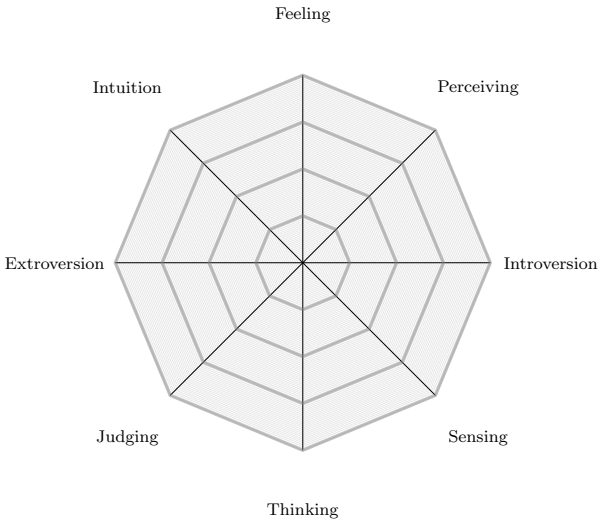


Figure 8.6: *A MBTI-Team Radar Template*

In this thesis, we investigated five software *teams*[7] from Simurg, and illustrated their trait structures by using an MBTI radar, we identified the team members and calculated their percentage for each dichotomy, e.g. a person in (E/I) scale can be found 20% extroverted, and therefore he is 80% introverted, in (S/N) scale if she is 40% sensor, and therefore she is 60% intuitive, etc. For another example, consider participant 48 from Table 8.10. On our scaling system this individual would be marked 70% on extroverted coordinate and 30% on the introverted scale. In this approach, even if someone was found extroverted, it allows us to see what percentage the introversion level is, to his or her extroversion degree. To the best of our knowledge, such a result has not previously been published.

8.6.2 Software Teams

Among the 213 industrial participants from the first case study, we interviewed a number of individuals to single out the available teams for this part of the study. After a careful consideration, 63 participants in a group of different soft-

[7]The names of the teams are concealed due to privacy reasons.

ware teams from the first case study were selected for this part. Apart from individuals qualifications and their work experience as a team, a major selection criteria is the team's availability to work with us during their hectic work life. Another selection criteria was based on the maturity of teams, which was defined by the team software process [328] throughout the software development organization. In light of this information, we selected and conducted the proposed card game on five software teams that can be identified as follows:

- ✔ *Triskele* - Research and development team of five people.
- ✔ *Camelot* - Software development team of twelve people.
- ✔ *Hector* - Software development team of eight people.
- ✔ *Finn* - Software development team of ten people.
- ✔ *Laran* - Software development team of sixteen people.

8.6.2.1 Team Triskele

Table 8.10 illustrates the personality characteristics of the team Triskele.

Participant ID	Job Title	E/I	S/N	T/F	J/P
P41	Project Manager	12%E 88%I	44%S 56%N	27%T 73%F	52%J 48%P
P46	Software Developer	70%E 30%I	40%S 60%N	25%T 75%F	33%J 67%P
P48	Software Specialist	11%E 89%I	12%S 88%N	4%T 96%F	33%J 67% P
P49	Software Specialist	67%E 33%I	25%S 75%N	32%T 68%F	55%J 45%P
P50	Software Developer	56%E 44%I	36%S 64%N	25%T 75%F	10%J 90%P

Table 8.10: *Team Triskele with roles versus members' traits*

It is apparent from Table 8.10 that there is a moderate balance in personality trait percentages for its team members. The percentage of intuition trait was very high which seems consistent to bring achievements in the innovative research effort. However, the most striking result to emerge from the data was that feeling trait was dominating the team, which shows there was higher agreement in the team than conflicts. In addition, perceiving was relatively higher than judging, which supports the fact found from the interviews that participants were inclined to use techniques from a set of agile practices instead of plan driven methodologies. Figure 8.7 below shows the team personality traits of team *Triskele*, scaled on an MBTI (team) radar format.

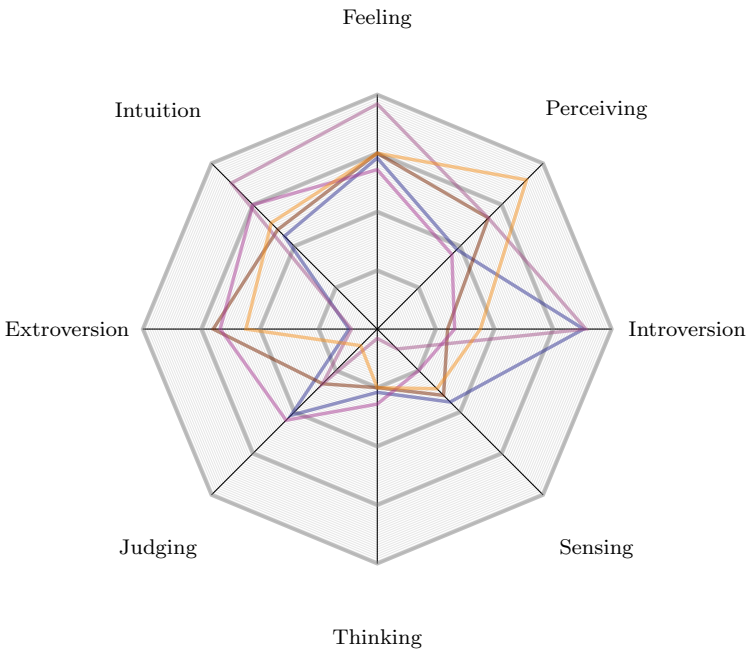


Figure 8.7: *Team Radar for Team Triskele*

8.6.2.2 Team Camelot

In the next team analysis, we calculated the personality traits of a team with 12 individuals. Table 8.11 shows the job titles versus some of the personality characteristics of the team Camelot. It is apparent from this table that individuals such as project managers, system analysts who may be socially active for their positions are more extroverted. Surprisingly perhaps, three of the team members were found 100% extroverted. As team Camelot was considered as one of the most productive teams of the development organization, the observed increase in extroversion could be attributed to the new tasks and activities of software development that require more socially interactive teams. For the other three dichotomies, the team showed a significant balance which is consistent with those of other studies and suggest that a team with a variety of attributes are expected to be more productive.

Figure 8.8 presents Table 8.11 data on a radar graph. From the graph, we can see that the extroversion reported significantly more introversion. For the other traits, most of the individuals are in the range between 25% and 75%, which

Participant ID	Job Title	E/I	S/N	T/F	J/P
P57	Project manager	100%E 0%I	67%S 33%N	52%T 48%F	39%J 61%P
P38	Software Developer	30%E 70%I	29%S 71%N	49%T 51%F	39%J 61%P
P40	Software Developer	19%E 81%I	39%S 61%N	34%T 66%F	49%J 51%P
P34	Software Developer	78%E 22%I	45%S 55%N	51%T 49%F	40%J 60%P
P47	Software Developer	36%E 64%I	66%S 34%N	57%T 44%F	65%J 35%P
P13	Software Developer	45%E 55%I	26%S 74%N	47%T 53%F	23%J 77%P
P45	Software Developer	55%E 45%I	43%S 57%N	40%T 60%F	40%J 60%P
P37	Software Tester	100%E 0%I	35%S 65%N	56%T 44%F	50%J 50%P
P44	Software Tester	100%E 0%I	28%S 72%N	59%T 31%F	61%J 39%P
P23	Software Tester	88%E 22%I	30%S 70%N	51%T 49%F	62%J 38%P
P24	Software Tester	55%E 45%I	56%S 44%N	38%T 62%F	28%J 72%P
P29	System Analyst	67%E 32%I	40%S 60%N	27%T 73%F	52%J 48%P

Table 8.11: *Team Camelot with roles versus members' traits*

indicates that the team has a good balance of personality traits.

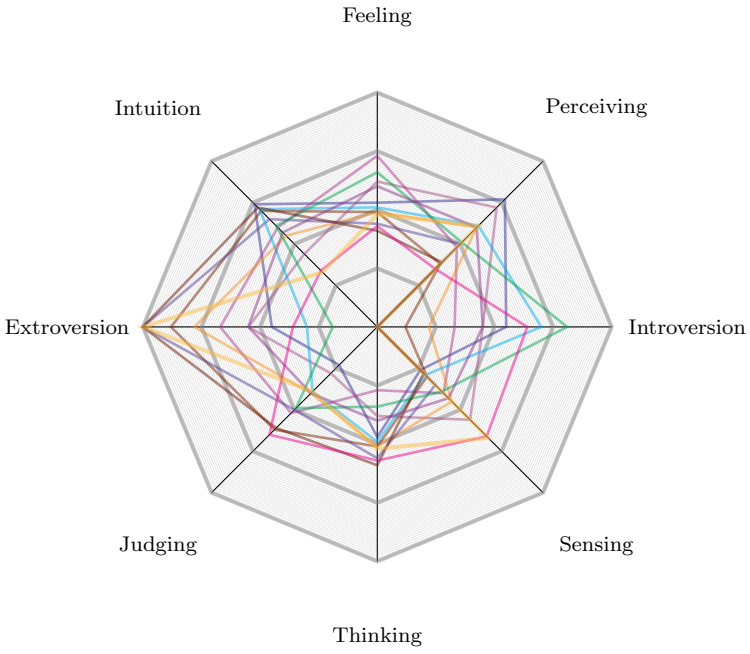


Figure 8.8: *Team Radar for Team Camelot*

8.6.2.3 Team Hector

Team Hector consists of four developers, a specialist, an analyst, and a manager. Table 8.12 shows participants id, job titles, and the percentage of personality traits found for the individuals from the team Hector. Similar to team Camelot, team Hector is also dominated with the extroversion trait, and there is a moderate balance on the other three dichotomous traits.

Participant ID	Job Title	E/I	S/N	T/F	J/P
P2	Software Developer	78%E 22%I	60%S 40%N	20%T 80%F	22%J 78%P
P3	Software Specialist	67%E 33%I	60%S 40%N	47%T 53%F	62%J 38%P
P8	Software Specialist	57%E 43%I	52%S 48%N	40%T 60%F	55%J 45% P
P9	Software Developer	65%E 35%I	48%S 52%N	39%T 61%F	56%J 44%P
P10	Software Developer	89%E 11%I	39%S 61%N	42%T 58%F	45%J 55%P
P12	System Analyst	66%E 34%I	55%S 45%N	32%T 68%F	56%J 44%P
P25	Software Developer	100%E 0%I	59%S 41%N	34%T 66%F	55%J 45%P
P62	Project Manager	79%E 21%I	42%S 58%N	27%T 73%F	23%J 77%P

Table 8.12: *Team Hector with roles versus personality traits*

Figure 8.9 presents Table 8.12 data on an MBTI radar graph.

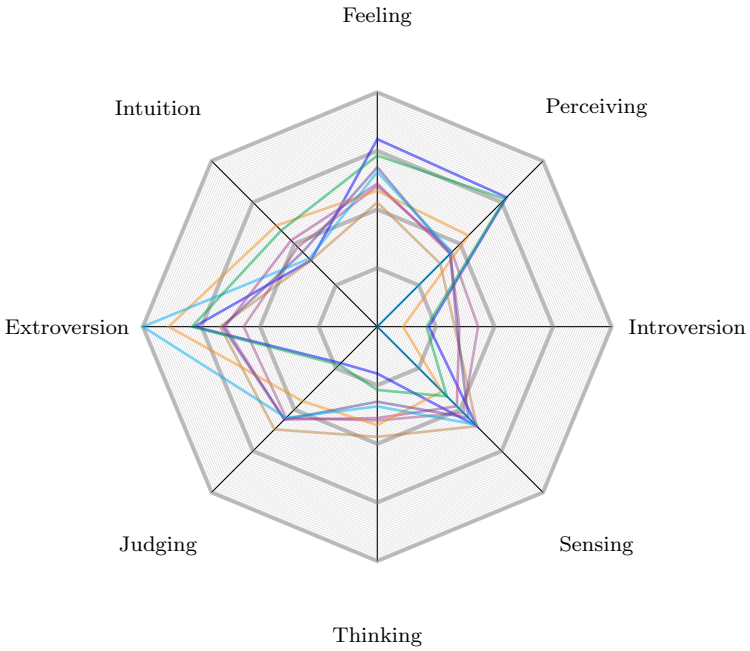


Figure 8.9: *Team Radar for Team Hector*

8.6.2.4 Team Finn

Table 8.13 provides the personality characteristics data for the team Finn. From this data, we can see that there were ten software practitioners with the roles such as software specialist, software developer, software testers, system analyst, and a project manager in the team. Similar to team Hector and Camelot, results indicate that team Finn also shows high extroversion, while the characteristics of team members (i.e. S/N, T/F, J/P) on the other traits seem to be equally distributed. The team also has more extroverted software developers than the other three teams.

Participant ID	Job Title	E/I	S/N	T/F	J/P
P5	Software Specialist	77%E 23%I	41%S 59%N	43%T 57%F	45%J 55%P
P6	Software Tester	76%E 24%I	33%S 67%N	59%T 41%F	67%J 33%P
P7	Software Specialist	91%E 9%I	41%S 59%N	51%T 49%F	39%J 61% P
P11	System Analyst	57%E 43%I	38%S 62%N	55%T 45%F	22%J 78%P
P15	Project Manager	59%E 41%I	28%S 72%N	36%T 64%F	73%J 27%P
P16	Software Developer	76%E 24%I	48%S 52%N	54%T 46%F	51%J 49%P
P17	Software Specialist	78%E 22%I	61%S 39%N	51%T 49%F	55%J 45%P
P18	Software Developer	91%E 9%I	57%S 43%N	32%T 68%F	34%J 66%P
P59	Software Developer	57%E 43%I	42%S 58%N	21%T 79%F	16%J 84%P
P61	Software Developer	48%E 52%I	34%S 66%N	59%T 41%F	40%J 60%P

Table 8.13: *Team Finn with roles versus personality traits*

Figure 8.10 shows the data in Table 8.13 on an MBTI radar graph.

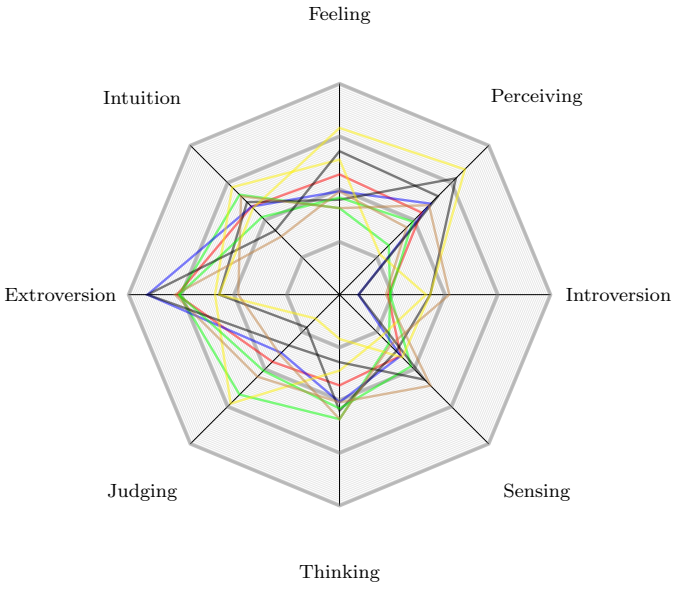


Figure 8.10: *Team Radar for Team Finn*

8.6.2.5 Team Laran

As can be seen from Table 8.14 that the team Laran has the role of software developer, software specialist, system analyst, team manager, software testers, and a project manager, and it is also highly populated with extroverted individuals.

Participant ID	Job Title	E/I	S/N	T/F	J/P
P1	Software Developer	78%E 22%I	62%S 38%N	59%T 41%F	38%J 62%P
P28	Software Specialist	55%E 45%I	38%S 62%N	69%T 31%F	55%J 45%P
P30	System Analyst	57%E 43%I	48%S 52%N	40%T 60%F	57%J 43% P
P31	Software Specialist	75%E 25%I	39%S 61%N	45%T 55%F	34%J 66%P
P32	Software Specialist	78%E 22%I	42%S 58%N	51%T 49%F	40%J 60%P
P33	Team Manager	49%E 51%I	63%S 37%N	33%T 67%F	57%J 43%P
P35	Software Tester	57%E 43%I	38%S 62%N	45%T 55%F	48%J 52%P
P36	Software Specialist	88%E 12%I	28%S 72%N	38%T 62%F	55%J 45%P
P39	Software Tester	68%E 32%I	46%S 54%N	53%T 47%F	79%J 21%P
P42	Software Developer	56%E 44%I	48%S 52%N	42%T 58%F	51%J 49%P
P43	Software Specialist	57%E 43%I	40%S 60%N	85%T 15%F	62%J 38%P
P51	Software Tester	34%E 66%I	40%S 60%N	32%T 68%F	39%J 61%P
P52	Software Specialist	75%E 25%I	63%S 37%N	17%T 83%F	56%J 44%P
P53	Software Developer	91%E 8%I	49%S 51%N	21%T 79%F	39%J 61%P
P54	Software Specialist	69%E 31%I	22%S 78%N	40%T 60%F	30%J 70%P
P55	Project Manager	100%E 0%I	43%S 57%N	48%T 52%F	33%J 67%P

Table 8.14: *Team Laran with roles versus personality traits*

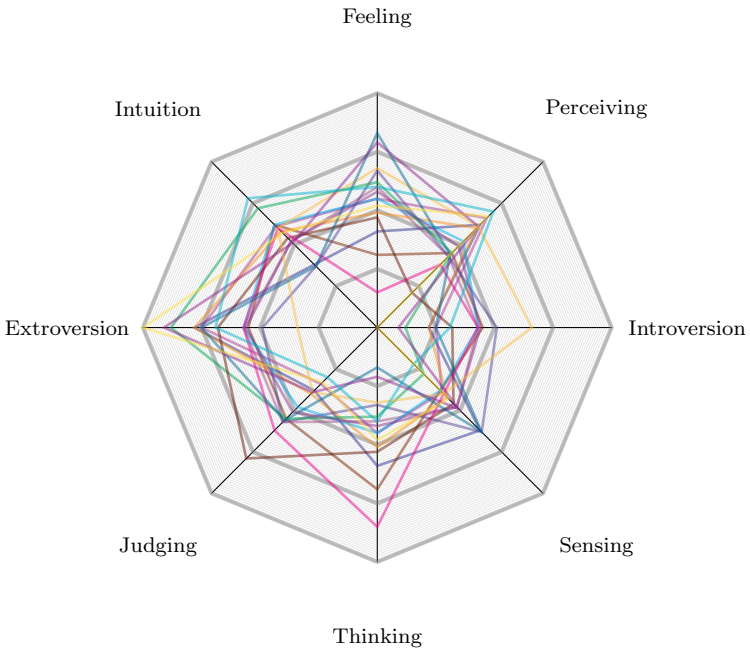


Figure 8.11: *Team Radar for Team Laran*

The single most striking observation to emerge from the data in comparison

with other teams was that this team has the greatest balance on the traits (i.e. S/N, T/F, J/P); the team average of participants is close to fifty percent, e.g. S/N: 42%, T/F:47%, J/P:46%, etc. Figure 8.11 shows the data in Table 8.14 on an MBTI radar graph.

8.6.2.6 A Brief Discussion about Findings

For the team Triskele, the results acquired from the data collected from the research and development team show that the team is relatively disconnected from social gatherings (e.g. they are not directly working with the customers), and there are no testers or a system analyst in their unit. As the team includes several researchers who should work with a limited number of people. Therefore, individuals who were selected for such teams are to be more inclined to introversion. This result was compatible with early days of software engineering landscapes when the teams were working in isolated environments, and practitioners were found to be more introverted. According to our results, this type of introversion was only observed in the *research and development* team, as we found all other four *cross-functional* software development teams were highly extroverted. For the traits (S/N, T/F, and J/P), all teams were in balance according to our analysis.

In contrast to team Triskele, other four teams namely Camelot, Hector, Finn, and Laran were found to be closer to the extroversion scale. The other four software teams were considered to be working in socially interactive settings. In particular, they are the teams which had members with strong focus on stakeholder engagement. In addition, the testers, system analysts and managers of the other four teams were found mostly extroverted. Lastly, for the teams working in isolated environments, the introverted members may be preferred. However, if the team needs to be in contact with customers, the extroverted people might work better, and furthermore for the traits (S/N, T/F, and J/P) team profiles should be constructed to balance the traits.

8.7 Case Study II: Threats to Validity

In this section, we discussed several potential threats we addressed earlier to deal with validity problems. To cope with problems of construct validity, first we prepared our questions parallel to the Keirsey temperament (sorter) questionnaire, and secondly we conducted two pilot studies with the same group of participants to identify flawed questions. Before the tests, we also asked the participants a short version of Keirsey's questionnaire to check the reliability of initial questions and to compare its understandability. After creating our game based MBTI instrument, which was expected to precisely measure the personality traits of individuals, we conduct semi-structural interviews to validate the questions of the survey instrument. All questions are constructed from the themes that are captured from interviews based on Keirsey temperament questionnaire. In addition, the preliminary results from the pilot studies were published so as to get early feedback from the academic peers. Before conducting an industrial evaluation of tests, the final version of cards was discussed with several experts from academia.

To deal with internal validity problems, first for the both pilot studies we used the same number of participants to avoid nonequivalent control group problem. Secondly, during the time between two pilot studies, there was no outside event that might be a threat to validity; therefore, we were able to prevent any history effect. Thirdly, participants were only exposed to the same test two times, therefore we did not observe a testing that might potentially affect or threaten the internal validity. Fourthly, we did not change our survey instrument (i.e. measuring device), which could potentiality be a threat to validity. Lastly, experimenter may consciously or unconsciously change the result of the study. To avoid this, we conducted the interview in a game form by which we motivate the participant to focus on the situational context rather than questions.

To avoid from the external validity problems, for the two pilot studies, we built *within-participants design* in which we used the same participants to take measures for the two attempts. One advantage of this work is that when the same participants contribute to the same conditions, it increases the chance of

having statistical significance [121]. Finally, from the reliability point of view, we detailed the data collection and documented the protocols that were used, and further we shared all components of the game-based survey instrument.

8.8 Chapter Summary

This chapter presented the results of the second case study, which was based on the game-base survey instrument designed for revealing the personality traits of software practitioners. We proposed a deck of (70) two-sided cards that describe hypothetical business situations based on real ones. We envisioned that the cards should be built from categorical themes based on "*the content analysis of the verbal behavior*" [329] of selected software practitioners. One side of each card shall consist of a keyword and a relevant picture to deepen the impression of the event or the situation. Moreover, events selected for the situational context should be placed with a similar pattern to the Keirsey temperament (sorter) questionnaire, i.e. available in [281]. The validation of cards by conducting a pilot study was described and the quantification process was explained in detail. Furthermore, an industrial evaluation was conducted using five software teams. The teams were identified regarding to its practitioners personality traits and the results were plotted using a MBTI radar graphs. The results of the analysis will be discussed in the next chapter of the thesis.

Part V

Discussions & Conclusions

Chapter 9

Discussions

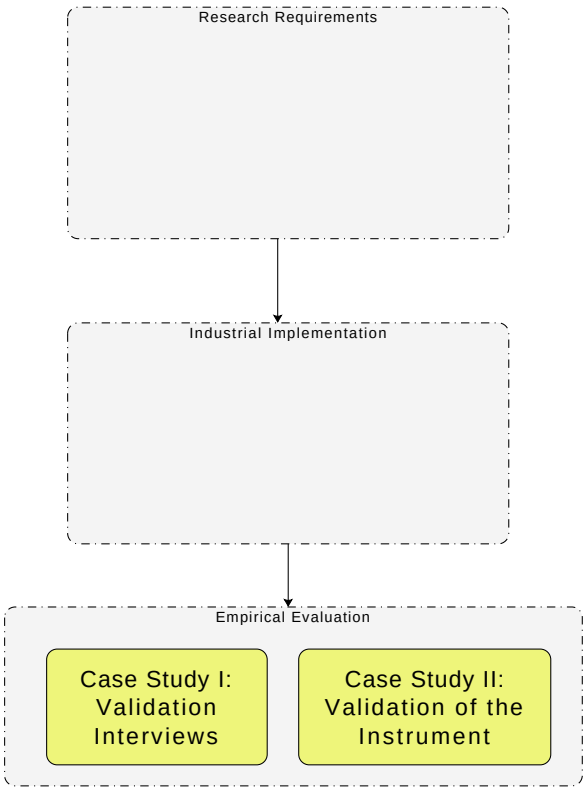


Figure 9.1: *A part of the conceptual overview of the research*

9.1 Introduction

This chapter discusses the two conducted case studies. First, we discuss the implications and some limitations of the constructed structural equation models, and the impact of roles on our productivity constructs. The validation interviews, which are a form of industrial discussions conducted in Simurg are also presented. In the following section, the interpretation of the results from the second case study as regards the software teams are discussed. The latter part of this chapter revisits the research questions and hypotheses from Chapter 1 to evaluate their truth or falsity.

9.2 Discussion of the Case Study I

In the first case study, we empirically evaluate the hypothesized relationships between the latent constructs and several factors that are affecting them (see Appendix C for survey data). The results confirm that productivity is highly associated with social productivity, and moderately associated with social capital. There is, however, a moderate correlation observed between social productivity and social capital. In particular, these empirical findings strongly support the notion that social factors dramatically influence software productivity. Returning to the social and organizational issues posed at the beginning of this study, it is now possible to state that most of the factors selected from the literature are affecting the productivity of a software development organization.

In general, the current findings add substantially to our understanding of the economic and social factors of productivity, which can be quantified using the SEM. In addition, this study is currently the most comprehensive (empirical) research that holds a significant value for industry and academia especially in that it develops a multifactor productivity measure. The multi-dimensional factors structure of the tripartite SEM model includes seven variables for measuring three of our constructs. To the authors' knowledge, this is also the first study of this nature to assess the implications of roles, team size and social capital on software development.

9.2.1 Validation Interviews

To understand how well we measure the productivity scale, one of the issues that emerge from these models is a need to evaluate them by a series of model validation interviews [144] with individuals from the management team of Simurg. We validated our models with the company by asking participants questions about the factors in the models and their opinion about the validity of these models such as "*What do you think about the company-based results we have found with SEM models?*", "*Do you think that any factor is missing or misrepresented in the productivity model? If so, which ones?*", "*Does your organization benefit from this new productivity perspective?*", "*Do you think these results may help the software development organization to improve their productivity?*"
As the management team discussed a series of simplified version of these models in a previous focus group study [330], they were delighted to examine the outcomes in this part of the work. In particular, they were mostly interested in Model IX, Model X, and Model XI. The interviewees were encouraged to comment on the relationships between the predictors, and latent constructs. Although some of the interviewees suggested some minor alterations about sorting the priority of factors, most of the participants had found these results consistent with respect to their expectations. The overall results of our structural models help the management team to discuss about the social factors, quantified latent constructs, and most importantly methods to improve their organizational productivity by using their implications.
Lastly, the results of this study indicate that software development organizations should be able to use our technique for measuring their organizational specific factors of productivity. An implication of this is the possibility of the management team's constructing a scale and identifying the causal relationships between indicators to see how causal ordering happens among these variables.

9.2.2 Limitations

The present study has a number of limitations. Firstly, the literature review on the factors of productivity is limited to the data we found in the literature.

Therefore, all SEM models are limited with the factors we were able to identify. Secondly, although we have nearly two hundred participants from an industrial company (Simurg), which can be considered as a substantial sample set in terms of software engineering studies to draw some empirical conclusions, we collected our data from a single software company, which should be tested with different settings for model comparison. Thirdly, there are possibilities for inadvertent sampling bias. Hence, to test the significance of common method error, models with more than two latent variables were tested for a single factor solution. Fourth, although this study benefits from an adequate sample size according to the SEM literature, we may extend our study to a greater sample size in a wider set of companies. To protect participants' confidentiality, participants were ensured their anonymity. Although there was no enforcement on the company level, we were able to obtain a substantial set of the data. Fifth, this work relies on a self-report measure. Therefore, we were unable to identify whether the same results can be observed with other data collection methods. Moreover, we conducted a cross-sectional study, i.e. our survey was conducted at a single point in time to obtain the variables and the constructs. Accordingly, the direction of causation and causal ordering cannot be determined by the collected data that does not provide significant substantiation for causality. In other words, all our models are based on correlational data that cannot be used to draw firm conclusions about the causal relationships. However, case studies and surveys were paired together as multiple methods to reduce the method bias.
There are only a few studies in the software engineering literature concerning the quantification of factors affecting the productivity of software development especially by using a sophisticated method like structural modeling. There are, for example, a SEM model for application development productivity [331], and a SEM model of feasibility evaluation and project success [332]. However, with a lack of evidence from other studies, caution must be applied, as the findings for now might not be transferable to all software development organizations.

9.3 Discussion of the Case Study II

The primary intent of the second case study is to design a game based instrument to identify personality types of individuals involved in a software development process. The instrument designed for personality tests aims to measure a fragment of human behavior. Therefore, evaluation of such an instrument should be validated by experimental investigations. To evaluate the reliability of questions, the card game was tested twice on sixteen participants in a six-month period (see Appendix E for both data sets). The contexts of these cards are built upon several business situations. We used grounded theory to analyze a series of interview data and compile keywords with different meanings for each situation that are identified and used in our card design process. Ultimately, the outcome of the game is the personality trait of an individual in an MBTI compatible scale. As a second step, we use these cards to reveal the personality types of 63 industrial practitioners (in a number of different teams) with a variety of different roles in a software development organization (see Appendix H). In addition, we use a questionnaire to identify the factors that potentially comprise the four Jungian personality types (in a 4-point Likert scale), which is based on the observation that are made in Simurg. This survey is used for calculating weights for each of the grounded factor potentially affecting the personality traits. Using this information, we calculate the average of weights for all questions (see Appendix G). We apply the results to our card game to discover the quantified form of personality types. In most industrial cases, the interviews reveal that participants who realized the personality differences among their teammates start questioning ways to improve their ability to communicate.

9.3.1 Validation of the Instrument

The results obtained from the preliminary analysis of the overall percentage of team traits are shown in Table 9.1

Figure 9.2 is a graphical representation of the data in Table 9.1 on an MBTI radar graph. The radar chart below shows the overall team traits where different

Team Name	E/I	S/N	T/F	J/P
Triskele	43%E 57%I	31%S 69%N	23%T 77%F	37%J 63%P
Camelot	64%E 36%I	42%S 58%N	47%T 53%F	46%J 54%P
Hector	75%E 25%I	52%S 48%N	35%T 65%F	47%J 53% P
Finn	71%E 29%I	42%S 58%N	46%T 52%F	44%J 56%P
Laran	68%E 32%I	44%S 56%N	45%T 55%F	48%J 52%P

Table 9.1: *Overall Percentages of Team Averages on Team Personality Traits*

colored drawings correspond with different team averages for the four traits. As can be seen from the figure, the overall extroversion is higher in all teams where the average of other three traits converge between the second and the third layer of spiderweb graph that should be interpreted as the existence of a team balance in terms of personality traits that were observed in most of the study cases.

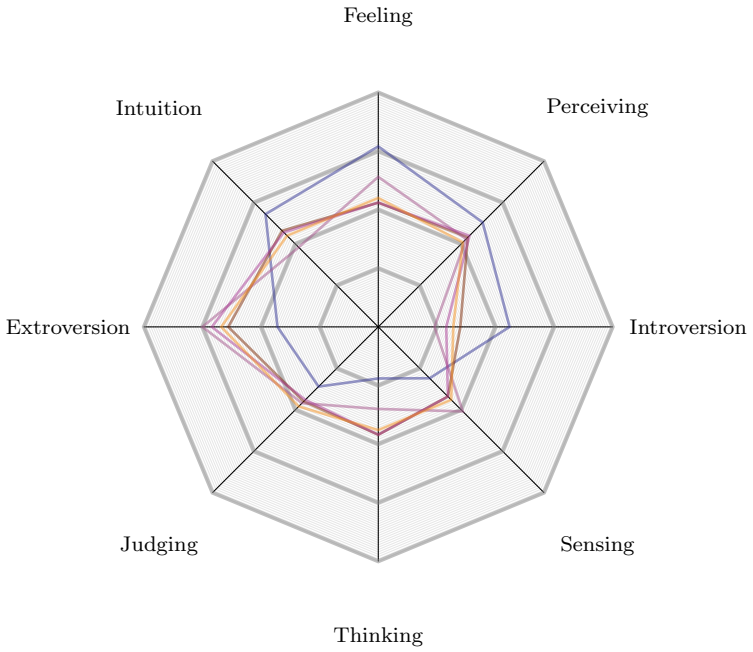


Figure 9.2: *The Averages of Personality Traits for All Teams*

These results, however, are consistent with those of recent studies and suggest that, because of the increasing demand for customer oriented activities, software business needs more sociable individuals, and therefore extroversion within the team members is likely to become more visible. This study confirms that some personality traits are more inclined to align themselves with specific development roles, e.g. the extroverted individuals are found to be software testers. For the traits (S/N, T/F, and J/P), the team shows a good balance among its

members who could facilitate the collective behavior (i.e. social cooperation) among its members.

ENFJ 12.7%					ENTJ 6.3%
INFJ 0%					INTJ 1.6%
ENFP 28.6%	ESFJ 7.9%	ESFP 6.3%	ESTP 1.6%	ESTJ 0%	ENTP 4.8%
INFP 20.6%	ISFJ 1.6%	ISFP 1.6%	ISTP 0%	ISTJ 1.6%	INTP 4.8%

Table 9.2: *Periodic Representation of the Percentage of Practitioners in the Sample*

Lastly, Table 9.2 represents a periodic classification of the personality types found in Simurg where 14 different personality types were identified using our game-based instrument. The trait that was found the most frequent was ENFP, where INFP, and ENFJ percentages of the population were found slightly higher than the other traits. From a temperament perspective, it is evident from the table that practitioners mostly populate the idealist and the rationalist columns. The periodic table offers a novel perspective for understanding the personality traits found in a software development organization, and it is rendered easier to envision the missing types of personalities if this approach is used for personality based team configurations.

9.3.2 Limitations

A number of limitations need to be considered. First, all kinds of personality tests build upon self report (i.e. subjective evidence), which inherently involves the possibility that participant report false choices. Although the game-based approach was likely to improve participants' motivation to reveal true preferences, to deal with this issue, we informed the participants that the results will be kept confidential, and announced that there is no wrong or right answers. Secondly, the first pilot assessment was conducted at a single point in time, which means it was conducted as a cross-sectional study. To address this problem, we conduct a replication study using the same participants and analyze the differences between the two findings.

9.4 Revisiting the Research Questions and Hypotheses

In this section, the research questions and hypotheses from Chapter 1 are discussed in the light of the two industrial case studies. This study has three main objectives; (i) build a productivity model based on the identified factors affecting it, (ii) investigate a set of team-based parameters and roles and team configurations over the productivity, (iii) explore the personality characteristics of software practitioners using a game-based personality type indicator. In light of these objectives, we created six research questions and three hypotheses.

> **RQ1:** *Can we quantify productivity by using a set of indicators and with the latent constructs (i.e. social capital and social productivity) that are potentially affecting productivity?*

To address the first research question, we propose a set of productivity models with a number of latent constructs by using statistical techniques. We consider software productivity as a latent variable (i.e. a construct not directly observed) in which a measurement scale for indication of the factors affecting software development productivity should be established. Such efforts can boost a potential for understanding the impact of these factors and be useful for improving the productivity of software development. This question has been answered by the industrial case study I, Chapter 7.6, which revealed positive correlations among several factors that are indicating the two of our latent constructs.

> **RQ2:** *Can a positive correlation between productivity, social productivity and social capital be measured for software development?*

To address the second research question, we presume that the software practitioners are intellectual workers who continuously collect and process a series of information (e.g. requirements, technologies) into a set of knowledge that actualizes as software artifacts. At the same time, the knowledge assets embedded inside the activities of a software development are used for generating an economic value. This value, however, should not only be determined by the outcome of the production process but also (i) as the human part of the

capital which encompasses the value added to the workers during the process and, (ii) as the social capital (i.e. *embedded resources in social networks* [249]), which is a form of capital captured by trustful social interrelations. In order to bridge the gap between formal and the social world of software practices, we propose a valuation of a software development productivity which should not only be realized by financial indicators in the form of the capital but also with its intellectual capital, and in particular in terms of both social productivity and capital. This question has been answered by a part of the industrial case study I, Chapter 7.8, where we built a tripartite SEM Model.

> **Hypothesis 1:** *There is a significant correlation between the factors affecting software development and the productivity of software development.*

In light of this argument, software productivity should be considered as a multidimensional concept that needs to be carried out from both sociological [4], and economical [16] perspectives. Productivity improvement is one of the main concerns of a software organization, which starts very early in any software development life cycle. It is a commonly used notion in software engineering, yet it is a concept difficult to define precisely. Although a generally accepted definition of productivity is lacking [10], it can basically be considered as the production rate or capacity of a process - something that agile software development often terms as the *project velocity*.

Moreover, we consider productivity as a value creation activity in a specific time period, which is, in general, hard to quantify. Boehm [16] reports the notion that *several factors are to be found in the attributes of people* and their interactions should be undertaken for the productivity improvement efforts. As previously mentioned, research has indicated that several factors such as size of a project, the development environment and the technologies (e.g. programming language) have a significant impact on software development productivity [17]. Based on the identified factors, we build several advanced SEM models, and test them with data collected from a software development organization. Next, we

analyze these models by using a series of statistical techniques that are embedded in the SEM methodology. The results of the empirical evidence presented in Chapter 7 indicate that several social and economic determinants extracted from the literature potentially affect the productivity of software development, and therefore we have found reasonably strong support for Hypothesis 1.
The second group of research questions focus on the relationship of roles and our latent constructs with respect to team based parameters such as actual team size, ideal team size and the work-experience of the practitioners.

> **Research Question 3:** *Can we observe a relationship between roles of software practitioners and the observed team productivity?*

According to the knowledge extracted from the conducted interviews, roles assigned to practitioners and size of a software team are found to be the two vitally important factors for improving the software team productivity. To observe their relationships, we ask the participants about a set of questions about team size and their effects on the team productivity. The analysis given in Chapter 7 confirms the observable relationships between roles and opinions of the software practitioners on team productivity.

> **Research Question 4:** *Is there any empirical relationship between social capital and identified variables to measure the variations in software team productivity?*

Moreover, we analyze the impact of software roles and the team-based factors affecting software development productivity. The statistically significant pairwise correlations among the identified pairs are shown in Table 7.9, which confirm that there is a negative association between social capital and the years software practitioners spend in Simurg.

> **Hypothesis 2:** *There is an observable relationship among the perceived team productivity, roles and our hypothetical (latent) constructs of software productivity*

The empirical evidence collected and presented in Chapter 7 confirm that there is only a negative relationship between one of the latent constructs (i.e. social capital) and the time a software practitioner spend in the software company. Although we found pair-wise relationships between some of the team-based parameters, we have not identified any other statistically significant relationships between other latent constructs and team based parameters. Thus, we have found moderate support for Hypothesis 2.

The classical vision of software production considers software practitioners who seek to maximize their utility alone. However, economic games offer a different perspective by perceiving the teams as an interacting ecology of networks. Therefore, there could also be long-term benefits to construct games for improving software development and team productivity. A game-based approach is found to be naturally motivating for maximizing the team productivity. To build a *personality-profiling instrument*, in the third part of this study, we propose two research questions:

> **Research Question 5:** *Can we reveal the personality traits of software practitioners by using a context specific, game-like profiling method?*

To answer this question, we first created a situation based, context dependent, MBTI compatible, game like personality test applicable to the software development practitioners and teams. To this end, we conducted several interviews based on the key context of psychometric questionnaires. By using the transcriptions organized through the grounded theory analysis and collected from industrial settings, context specific questions were prepared. The construction process can be seen in Chapter 8, Section 8.2, the rules for the game can be found in Section 8.3.

> **Research Question 6:** *Can we build a visualization instrument to illustrate software team personality types?*

To address these questions, first we conducted an industrial case study with the questionnaire that is statistically validated in Chapter 8, Section 8.4. Secondly,

to explore different team combinations, five teams were selected from Simurg for personality type investigation, and the results were illustrated in a special form of radar graph, which we termed as MBTI radar. The team details are illustrated in Chapter 8, Section 8.5.2

> **Hypothesis 3:** *Personality characteristics of individuals in software development teams can be revealed and illustrated by using a context specific game-based profiling technique.*

The empirical evidence collected from five different software teams are presented in Chapter 8. We can confirm that extroversion is a dominant trait in the observed software teams. All of the software testers interviewed except one are found extroverted while software developers seem to have both introverted and extroverted characteristics. Project managers, on the other hand, rated usually high in extroversion scale. It is therefore likely that testers and project managers are observed to be gravitating towards extroversion. For the most of the participants, however, the three other personality traits (S/N, T/F, J/P) are mostly in balance among the teams (see Figure 9.2) and further details can be found in Chapter 8, Section 8.5.2. In brief, we have found moderate support for Hypothesis 3.

9.5 Chapter Summary

This chapter showed the discussion over the two industrial case studies, their limitations and validation techniques. The research questions and research hypothesis in Chapter 1 were discussed here to extend our understanding of the subject matter. The next chapter will give an overview of the conclusions of the both industrial case studies, contributions, and future work. The main objective of the next chapter is to present the conclusions of this research. Next, the researchers also will present the research contribution and examine the limitations of the present study. Finally, some future research possibilities will be presented which could build upon the present research study.

Chapter 10

Conclusions and Future Work

10.1 Introduction

The goal of this chapter is to derive a set of conclusions from the two industrial case studies. It starts with our model proposition for software development productivity and gives some conclusions for the first case study. Further, it continues by giving some details about game based personality analysis, and conclude the second industrial case study. In the final section of this chapter, overall research implications and future directions for the research are discussed.

10.2 Industrial Case Study I

The following conclusions can be drawn from the present study. Our empirical evaluation shows conclusively that there is a significant positive correlation among the latent constructs, all of which can be explained by the identified factors. Based on these correlations, the empirical findings in this study provide a new understanding of productivity in terms of social productivity and social capital. Therefore, it is evident that there is a relationship between social capital and social productivity; while social productivity has more impact on productivity. With regard to practical implications, we conclude that social capital and its transformation to social productivity deserve more attention because this process has the potential to improve software development

productivity. The team factors with respect to the roles of the participants, which are performed in the second part of the analysis promotes that there were significant correlations between several team based variables and our latent constructs. For example, individuals who are more experienced in the software development organization were observed to work in larger sized teams and they are inclined to think that the social capital was of less importance.

Given such theory about the connections between productivity and factors affecting it, it is possible to interpret that the relationship between productivity and social capital is mediated by social productivity (see Chapter 7, Figure 7.12). Taken together, these findings enhance the understanding of the management team of a company about productivity factors from the software organization's point of view. They offer a useful method of quantification for the latent constructs. However, it is recommended that further research be undertaken to examine the associations among productivity constructs.

This study makes marked contributions to the software productivity literature. As previously mentioned, Jones reported that there are yet no effective measurement ways found for software development productivity [10]. Therefore, this study can be considered as a first attempt to measure the software development productivity with the factors found from the literature and further evaluate the results from an industrial perspective. Although several previous studies mentioned the importance of social aspects of software development, the implications of social capital on software development productivity has not been deeply investigated. To bridge this gap, we build several models, and introduce the notion of social productivity of software development and link it with both social capital and software development productivity. Taken together, our approach could assist the management team of a software development organization to identify and quantify company-specific factors to improve organizational productivity.

Future research should therefore concentrate on the investigation of validity of the latent constructs with samples from alternative software organizations in several different settings. Such a study would be of great value for understand-

ing *the productivity dynamics* of a software development organization and for managing the factors that are potentially affecting the structure of an organization.

10.3 Industrial Case Study II

Software projects face several challenges in their dynamically changing organizational environments. These challenges affect software practitioners who have a set of distinctive personality types. The members of the software teams socially interact to perform a series of tasks or assignments in a software development project. In fact, one of the key components of success in a software development organization is selection of the right employee or a team for the right tasks. From a technical viewpoint, skills of the individuals should match with the required talents and experience. However, to improve software productivity, the social aspects such as individuals' compatibility in a team has emerged as a research interest with a focus on personality traits over software team configuration [333], which directly affects the quality of knowledge exchange among the team members. It is therefore not surprising to discover that several researchers in the field of software engineering have focused on the effects of personality types on the software development process and organizational performance [334–336].

The theory of games and its implementation on software development organizations can provide a way to explore the effects of social structures on team composition, where we can use this information for creating better team configurations. An economic mechanism involves designing the rules for the economic activities that govern the social interactions of the participants. These rules, for example, can be designed to motivate individuals by stimulating them to behave in a certain manner, and to achieve certain economic or social outcomes. Finally, a mechanism establishes the fabric between the actions of individuals and social landscapes of software organizations. We suggest that, a mechanism enables us to maximize the economic and social outputs of the software development effort - through modeling the structure of software teams and fur-

ther envisioning a software development organization. We aim to establish a structural improvement for a software team based on the fact that the quality of organizational production relies on the structure of the organization [337]. Based on the selectable parameters for desired goals or given objectives, we define a mechanism as *a game form*. It is built on several inputs from individuals in order to produce the desired outputs. Our goal is to dynamically portray the personality traits of an individual for designing an optimal team structure using this game form. The game is designed to motivate individuals to reveal their personality types to assist building more effective team configuration.

A software team comprised of participants with several different characteristics, who are bound to frequently interact. Consequently, compatibility of their personalities becomes an important concern for the team success. Over the last decade, the personality tests have become a standard tool for assessing individuals in a typical hiring process [338]. Regardless of being agile or traditional, a software team is formed to respond to the key challenges such as the increased diversity in activities and the required interactions in a software development process. During the interviews, we observed a relationship between the agile proponents and the practitioners with perceiving trait. Nearly all individuals we interviewed have a significant inclination to be agile, and are found to be in the perceiving trait. Thus, we can conclude that our approach is also useful for understanding the team members compatibility or tendency to use agile or plan driven methodologies based on the selections between socio-type (J) and (P). Consequently, *situational context cards* shall provide a mechanism for balancing agility and discipline from a team configuration perspective to some extend.

In the second case study, we have demonstrated that a game-based approach is relatively easier to reveal the true personality types of individuals than a paper-based alternative. Such an approach should further improve our ability to build or configure more effective software teams or perhaps in the process of integrating a new member to software team structure. Our approach contributes to a software development process by illustrating a team's personality structure on team radar. In practice, it will be useful not only for building

software development teams but also having a favorable selection from a set of individuals with the same skill set with different personalities. In job applications for example, the approach might be useful for choosing the most suitable person from a group of candidates by using an observable variability in aspects of their behavior. Moreover, we believe that our approach has the potential to supersede the paper based MBTI tests, particularly developed for software development organizations. However, our findings may not be extrapolated to all software companies. Further studies, which take these outputs into account, will need to be undertaken.

10.4 Research Contributions

The key contributions of this research are classified into two aspects; theoretical and practical. From a theoretical point of view, to the best of our knowledge, this is the first published research study that considers software team configuration as a mechanism design problem and proposes a game-based team configuration model using the personality traits of individuals. This model was built to accommodate a rigorous way for understanding effective team configurations. From a practical point of view, it was applied to reveal the software development team configurations with respect to practitioners' personality types. This study provides strong support for the conceptual premise that a game based approach can be used successfully to reveal the personality characteristics of software practitioners.

The second theoretical contribution is to build a foundation for understanding the social and value dynamics of software development. Before building a set of empirical models of productivity, we define the following findings to make several contributions to the current literature. Firstly, we formulate software development productivity as a latent construct to model the software development process based on the economic and social factors found from the literature, which are potentially affecting it. In order to build a safe passage from the technical world of information artifacts into the factors affecting the social world of software development, we select a known model of social capital. To understand

the relationships between these two constructs, secondly this study introduces the notion of social productivity for software development.

From a practical point of view, the empirical relationships among these factors and the constructed models were validated by an industrial case study. In an effort to link these hypothetical models and relationships, eleven structural equation models are iteratively designed so as to demonstrate the relationship among these social constructs and their affecting factors. By combining quantitative and qualitative aspects inherently, SEM models are capable of measuring the relations between the constructs and the selected factors by using empirical observation from the field. These findings, while preliminary, confirm a dynamic relationship between the selected social constructs of productivity, social productivity, and social capital. These relations can be measured for a software development organization to quantify a customized model that could be useful for improving organizational success.

The third theoretical contribution is that we visually extend the team-based information architecture and improve the representation skills of software management. First, to enable managers to select and tailor not only a process but also roles for their activities of software development, a visual summary of roles in different development methodologies is made. From an empirical point of view, this chart helps researchers to reveal the actual relationships between project roles and previously identified social constructs. Secondly, a novel periodic table approach is introduced, which can be used to visually illustrate the percentages of personality characteristics of software organization as a whole. Empirically, the identification of the organizational sample outlined the defined characteristics. Thirdly, a special form of radar chart, MBTI-Team Radar is conceptualized. Then, it is used to visualize the collected data regarding to personality characteristics of software practitioners as a whole team. Consequently, all of these techniques provide software managers with a way of visualizing the impact of their team designs, or their personnel selections. It also helps them to improve software team success, and therefore potential return on the investment.

10.5 Recommendations for Future Work

In this final section, we discuss some of the research possibilities for future work regarding how the next level efforts of the study should continue. These efforts can best be treated under three headings: theoretical, empirical, and representational improvements.

Firstly, the game theoretical model can be improved by designing new games. For example, in future investigations it might be possible to conduct action research in order to investigate the social interaction of software development teams by using the prisoners' dilemma framework, which is one of the most significant areas of research in the field of game theory. Such a study can be helpful for revealing the problems of coupling of two practitioners (e.g. pair programming or peer reviews) that should detail relationships between the software practitioners with distinctive personality traits.

Additionally, in order to improve the performance of our game-based method, more empirical case studies should be performed by using the game approach for revealing personality types of individuals in different software organizations. For example, new cards may be designed and the game should be balanced regarding gained experience with more team-based sessions.

Secondly, we confirm that understanding how factors impact the software development productivity will directly improve the economic viability of a software system. However, this knowledge can be enhanced by further empirical studies. Theoretically, the structural equation models could likewise be improved by gathering more factors from the literature for all of the three latent constructs. Practically, these models should also be empirically tested on other software development organizations for more generalizable results. Therefore, more research on this topic needs to be undertaken before the connection between the latent constructs and the factors affecting them are more clearly understood.

The experience from the field suggests that empirical analyses alone may not be enough to convince the management for these types of improvements. Equally important, several visualization methods have emerged naturally as a way of presenting our findings during this study. This researcher believes that the

results of this thesis should be properly illustrated; therefore, further research should be done to investigate more representation techniques for the gathered data and initial findings. Future studies on the visualization techniques to improve the periodic table form or the role charts are therefore highly recommended.

Bibliography

[1] R. Conradi and A. Fuggetta, "Improving software process improvement," *IEEE Software*, vol. 19, no. 4, pp. 92–99, 2002.

[2] R. L. Glass, *Facts and Fallacies of Software Engineering*. Addison-Wesley Professional, 2002.

[3] S. T. Acuna, N. Juristo, A. M. Moreno, and A. Mon, *A Software Process Model Handbook for Incorporating People's Capabilities*. Springer-Verlag, 2005.

[4] T. DeMarco and T. Lister, *Peopleware: productive projects and teams*. Dorset House Publishing Company, 1999.

[5] Y. Dittrich, C. Floyd, and R. Klischewski, *Social thinking-software practice*. The MIT Press, 2002.

[6] M. Grechanik and D. E. Perry, "Analyzing software development as a noncooperative game," in *IEE Seminar Digests*, vol. 29, 2004.

[7] H. Van Vliet, "Editorial: Signs of a thriving journal," *Journal of Systems and Software*, vol. 86, no. 1, p. 1, 2013.

[8] R. Charette, "Why software fails," *IEEE Spectrum*, vol. 42, no. 9, pp. 42–49, 2005.

[9] D. Hartmann, "Interview: Jim johnson of the standish group," *Infoqueue, Aug*, vol. 25, 2006.

[10] C. Jones, *Software Engineering Best Practices: Lessons from Successful Projects in the Top Companies.* McGraw-Hill Osborne Media, 2009.

[11] J. E. Tomayko and O. Hazzan, *Human Aspects of Software Engineering.* Firewall Media, Dec. 2005.

[12] G. M. Weinberg, *The psychology of computer programming.* Van Nostrand Reinhold New York, 1971.

[13] H. Robinson and H. Sharp, "Collaboration, communication and coordination in agile software development practice," in *Collaborative Software Engineering.* Berlin Heidelberg: Springer, 2010, pp. 93–108.

[14] S. Biffl, A. Aurum, B. Boehm, H. Erdogmus, and P. Grunbacher, *Value-based software engineering.* Springer, 2005.

[15] StandishGroup, "The chaos report," Available on-line at http://www.projectsmart.co.uk/docs/chaos-report.pdf, 1995.

[16] B. Boehm, *Software Engineering Economics.* Prentice Hall, Nov. 1981.

[17] R. Selby, *Software engineering: Barry W. Boehm's lifetime contributions to software development, management, and research.* Wiley-IEEE Computer Society Pr, 2007.

[18] K. Sullivan, P. Chalasani, and S. Jha, "Software design decisions as real options," University of Virginia, Tech. Rep., 1997.

[19] H. Erdogmus, B. Boehm, W. Harrison, D. Reifer, and K. Sullivan, "Software engineering economics: background, current practices, and future directions," in *Proceedings of the 24th International Conference on Software Engineering.* ACM, 2002, pp. 683–684.

[20] B. Boehm, "Software engineering economics," *IEEE Transactions on Software Engineering*, no. 1, pp. 4–21, 1984.

[21] J. D. Blackburn, G. D. Scudder, and L. N. Van Wassenhove, "Improving speed and productivity of software development: A global survey of software developers," *IEEE Transactions on Software Engineering*, vol. 22, no. 11, pp. 875–885, 1996.

[22] B. Boehm, "Value-based software engineering: reinventing," *ACM SIGSOFT Software Engineering Notes*, vol. 28, no. 2, p. 3, 2003.

[23] M. Halling, S. Biffl, and P. Grunbacher, "The role of valuation in value-based software engineering," in *Proceedings of the 6 th International Workshop on Economics-Driven Software Engineering Research (EDSER-6)*, 2004.

[24] S. T. Acuna, M. Gomez, and N. Juristo, "How do personality, team processes and task characteristics relate to job satisfaction and software quality?" *Information and Software Technology*, vol. 51, no. 3, pp. 627 – 639, 2009.

[25] M. André, M. Baldoquín, and S. Acuña, "Formal model for assigning human resources to teams in software projects," *Information and Software Technology*, vol. 53, no. 3, pp. 259–275, 2011.

[26] C. Warhurst, I. Grugulis, and E. Keep, *The skills that matter*. Palgrave Macmillan Houndmills, 2004.

[27] N. Hanna, *Enabling Enterprise Transformation: Business and Grassroots Innovation for the Knowledge Economy*. Springer Verlag, 2009.

[28] F. Tsui, *Managing software projects*. Jones & Bartlett Learning, 2004.

[29] S. Ryan and R. V. O'Connor, "Development of a team measure for tacit knowledge in software development teams," *Journal of Systems and Software*, vol. 82, no. 2, pp. 229–240, Feb. 2009.

[30] A. Kuper and J. Kuper, *The social science encyclopedia*. Routledge/Thoemms Press, 1985.

[31] K. Leyton-Brown and Y. Shoham, *Essentials of game theory: A concise multidisciplinary introduction.* Morgan & Claypool Publishers, 2008.

[32] N. Nisan, *Algorithmic game theory.* Cambridge Univ Pr, 2007.

[33] F. Deek, J. McHugh, and O. Eljabiri, *Strategic software engineering: an interdisciplinary approach.* CRC Press, 2005.

[34] T. Dingsoyr, T. Dyba, and N. Moe, *Agile Software Development: Current Research and Future Directions.* Springer Publishing Company, Jun. 2010.

[35] J. McGonigal, *Reality is broken: Why games make us better and how they can change the world.* Penguin Pr, 2011.

[36] G. Matthews, I. J. Deary, and M. C. Whiteman, *Personality Traits*, 3rd ed. Cambridge University Press, Nov. 2009.

[37] D. Mayer and A. Stalnaker, "Selection and evaluation of computer personnel-the research history of SIG/CPR," in *Proceedings of the 1968 23rd ACM national conference.* ACM, 1968, pp. 657–670.

[38] N. Kerth, J. Coplien, and J. Weinberg, "Call for the rational use of personality indicators," *Computer*, vol. 31, no. 1, pp. 146–147, Jan. 1998.

[39] P. Kline, *The handbook of psychological testing.* Psychology Press, 2000.

[40] E. Kaluzniacky, *Managing psychological factors in information systems work: An orientation to emotional intelligence.* Information Science Publishing, 2004.

[41] W. Humphrey, *Managing the Software Process.* Addison-Wesley, 1990.

[42] A. Fuggetta and A. L. Wolf, *Software process.* J. Wiley, 1996.

[43] I. Sommerville, *Software Engineering (9th Edition).* Addison Wesley, Mar. 2009.

[44] P. Feiler and W. Humphrey, "Software process development and enactment: Concepts and definitions," in *Software Process, 1993. Continuous Software Process Improvement, Second International Conference on the.* IEEE, 1993, pp. 28–40.

[45] H. Erdogmus, "Essentials of software process," *IEEE Software*, vol. 25, no. 4, pp. 4–7, 2008.

[46] ISO/IEC, *Amendment to ISO/IEC 12207-2008 - Systems and software engineering Software life cycle processes*, 2008.

[47] R. V. O'Connor, "Human aspects of information technology development," *International Journal of Technology, Policy and Management*, vol. Vol. 8, No. 1, 2008.

[48] S. Zahran, *Software Process Improvement: Practical Guidelines for Business Success.* Addison Wesley, 1998.

[49] J. R. Persse, *Process Improvement Essentials.* O'Reilly Media, Inc., Sep. 2006.

[50] D. Wastell, "The Human Dimension of the Software Process," *Software Process: Principles, Methodology, and Technology*, pp. 165–199, 1999.

[51] A. Fuggetta, "Software process: a roadmap," in *ICSE '00: Proceedings of the Conference on The Future of Software Engineering.* ACM, 2000, pp. 25–34.

[52] B. Unhelkar, *Practical Object Oriented Analysis.* Thomson Publishing, Mar. 2005.

[53] W. Royce, "Managing the development of large software systems," in *Proceedings of IEEE Wescon*, vol. 26, 1970.

[54] V. R. Basili and A. J. Turner, "Iterative enhancement: A practical technique for software development," *IEEE Transactions on Software Engineering*, vol. 4, pp. 390–396, 1975.

[55] G. Pomberger, W. R. Bischofberger, D. Kolb, W. Pree, and H. Schlemm, "Prototyping-Oriented software development - concepts and tools," *Structured Programming*, vol. 12, no. 1, pp. 43–60, 1991.

[56] B. Boehm, "A spiral model of software development and enhancement," *Computer*, vol. 21, no. 5, pp. 61–72, 1988.

[57] B. Boehm, A. Egyed, J. Kwan, D. Port, A. Shah, and R. Madachy, "Using the WinWin spiral model: A case study," *Computer*, vol. 31, no. 7, pp. 33–44, 1998.

[58] K. Beck, *Extreme programming explained*. Addison-Wesley, 2000.

[59] K. Schwaber, "Agile project management with scrum," 2009.

[60] S. R. Palmer and J. M. Felsing, *A practical guide to feature-driven development*. Prentice Hall PTR, Feb. 2002.

[61] B. Mutafelija and H. Stromberg, *Process Improvement with CMMI v1. 2 and ISO Standards*. Auerbach Publications, 2008.

[62] R. Pressman, *Software engineering: a practitioner's approach*. McGraw-Hill New York, 2010.

[63] B. Boehm and R. Ross, "Theory-W software project management principles and examples," *IEEE Transactions on Software Engineering*, vol. 15, no. 7, pp. 902–916, 1989.

[64] B. Boehm and P. Bose, "A collaborative spiral software process model based on theory w," 1994.

[65] B. Boehm, "Anchoring the software process," vol. 13, no. 4, pp. 73–82, 1996.

[66] B. Hansen, J. Rose, and G. Tjornehoj, "Prescription, description, reflection: the shape of the software process improvement field," *International Journal of Information Management*, vol. 24, no. 6, pp. 457–472, Dec. 2004.

[67] T. Dyba, T. Dingsyr, and N. B. Moe, *Process Improvement in Practice: A Handbook for It Companies (The Kluwer International Series in Software Engineering, 9)*. Kluwer Academic Publishers, 2004.

[68] R. Kittler, M. Kocifaj, and S. Darula, *Daylight science and daylighting technology*. Springer, 2011.

[69] W. Shewhart and W. Deming, *Statistical method from the viewpoint of quality control*. Dover Publications, 1986.

[70] M. B. Chrissis, M. Konrad, and S. Shrum, *CMMI Guidlines for Process Integration and Product Improvement*. Addison-Wesley Longman Publishing Co., Inc., 2003.

[71] W. S. Humphrey and M. I. Kellner, "Software process modeling: principles of entity process models," in *ICSE '89: Proceedings of the 11th international conference on Software engineering*. ACM, 1989, pp. 331–342.

[72] D. Hoyle, *ISO 9000 quality systems handbook*. Butterworth-Heinemann, 2006.

[73] K. E. Emam, J. Drouin, W. Melo, and A. Dorling, *SPICE: The Theory and Practice of Software Process Improvement and Capability Determination*. Wiley-IEEE Computer Society Pr, Oct. 1997.

[74] M. Paulk, B. Curtis, M. Chrissis, and C. Weber, "Capability maturity model for software, version 1.1. software engineering institute," CMU/SEI-93-TR-24, DTIC Number ADA263403, Tech. Rep., 1993.

[75] SPICE, "Software process improvement and capability dEtermination," http://www.sqi.gu.edu.au/SPICE/, 2007.

[76] R. Singh, "International Standard ISO/IEC 12207 software life cycle processes," *Software Process Improvement and Practice*, vol. 2, no. 1, pp. 35–50, 1996.

[77] S. Ambler, "Lessons in agility from internet-based development," *Software, IEEE*, vol. 19, no. 2, pp. 66–73, 2002.

[78] W. S. Humphrey, *A Discipline for Software Engineering*, 1st ed. Addison-Wesley Professional, 1995.

[79] J. Favaro, "Managing requirements for business value," *IEEE Software*, pp. 15–17, 2002.

[80] A. Cockburn, *Agile Software Development*. Addison-Wesley Professional, Dec. 2001.

[81] M. Fowler and J. Highsmith, "The agile manifesto," *Software Development Magazine*, vol. 9, no. 8, pp. 28–35, 2001.

[82] B. Boehm and R. Turner, *Balancing Agility and Discipline: A Guide for the Perplexed*, 1st ed. Addison-Wesley Professional, Aug. 2003.

[83] J. Highsmith and A. Cockburn, "Agile software development: the business of innovation," *Computer*, vol. 34, no. 9, pp. 120–127, 2002.

[84] G. Miller, "The characteristics of agile software processes," in *Proceedings of the 39th International Conference and Exhibition on Technology of Object-Oriented Languages and Systems (TOOLS39)*. IEEE Computer Society, 2001, p. 385.

[85] K. Bittner and I. Spence, *Managing iterative software development projects*. Addison-Wesley, 2007.

[86] H. Coolican, *Research methods and statistics in psychology*. Hodder Education, 2009.

[87] L. Blaxter, C. Hughes, and M. Tight, *How to research*. Open University Press, 2010.

[88] L. Given, *The Sage encyclopedia of qualitative research methods*. Sage Publications, 2008.

[89] C. Goulding, *Grounded theory: A practical guide for management, business and market researchers.* SAGE Publications, 2002.

[90] S. VanderStoep and D. Johnson, *Research methods for everyday life: Blending qualitative and quantitative approaches.* Jossey-Bass, 2008, vol. 24.

[91] R. Galliers, "Choosing appropriate information systems research approaches: a revised taxonomy," 1992.

[92] R. Glass, I. Vessey, and V. Ramesh, "Research in software engineering: an analysis of the literature," *Information and Software Technology*, vol. 44, no. 8, pp. 491–506, 2002.

[93] C. Dawson, *Practical research methods: a user-friendly guide to mastering research techniques and projects.* How To Books Ltd, 2002.

[94] R. Stake, *Qualitative Research: Studying How Things Work.* The Guilford Press, 2010.

[95] A. Holliday, *Doing and writing qualitative research.* Sage Publications, 2007.

[96] S. Hesse-Biber and P. Leavy, *The practice of qualitative research.* Sage Publications, 2010.

[97] J. Maxwell, *Qualitative research design: An interactive approach.* Sage Publications, 2004.

[98] D. Muijs, *Doing quantitative research in education with SPSS.* Sage Publications, 2010.

[99] M. Balnaves and P. Caputi, *Introduction to quantitative research methods: An investigative approach.* Sage Publications, 2001.

[100] J. Creswell, *Research design: Qualitative, quantitative, and mixed methods approaches.* Sage Publications, 2009.

[101] P. Verschuren, "Case study as a research strategy: some ambiguities and opportunities," *International Journal of Social Research Methodology*, vol. 6, no. 2, pp. 121–139, 2003.

[102] A. Bryman, "Integrating quantitative and qualitative research: how is it done?" *Qualitative research*, vol. 6, no. 1, pp. 97–113, 2006.

[103] T. Cook and D. Campbell, "Quasi-experimental design and analysis issues for field settings," *Chicago: Rand Mc-Nally College Publishing Co*, 1979.

[104] N. Denzin, *The research act: A theoretical introduction to sociological methods*. Aldine De Gruyter, 2009.

[105] T. Jick, "Mixing qualitative and quantitative methods: Triangulation in action," *Administrative science quarterly*, pp. 602–611, 1979.

[106] A. Tashakkori and C. Teddlie, *Handbook of mixed methods in social & behavioral research*. Sage Publications, 2002.

[107] S. Andrew and E. Halcomb, *Mixed Methods Research for Nursing and the Health Sciences*. Wiley-Blackwell, 2009.

[108] R. Panneerselvam, *Research methodology*. PHI Learning Pvt. Ltd., 2004.

[109] R. Thietart, *Doing management research: a comprehensive guide*. Sage Publications, 2001.

[110] C. Seaman, "Qualitative methods in empirical studies of software engineering," *IEEE Transactions on Software Engineering*, vol. 25, no. 4, pp. 557–572, 1999.

[111] R. Hall, *Applied social research: planning, designing and conducting real-world research*. Macmillan Education AU, 2008.

[112] H. Holz, A. Applin, B. Haberman, D. Joyce, H. Purchase, and C. Reed, "Research methods in computing: What are they, and how should we teach them?" in *ACM SIGCSE Bulletin*, vol. 38, no. 4. ACM, 2006, pp. 96–114.

[113] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical Software Engineering*, vol. 14, no. 2, pp. 131–164, 2009.

[114] J. Creswell, *Research Design: Qualitative and Quantitative Approaches*. SAGE Publications, 1994.

[115] B. Berg, *Qualitative research methods for the social sciences*. Boston, MA, USA: Pearson Education, 2004.

[116] B. Oates, *Researching information systems and computing*. Sage Publications, 2005.

[117] R. Yin, *Case study research: Design and methods*. Sage Publications, 2008, vol. 5.

[118] B. Kitchenham, "Evaluating software engineering methods and tool part 1: The evaluation context and evaluation methods," *ACM SIGSOFT Software Engineering Notes*, vol. 21, no. 1, pp. 11–14, 1996.

[119] D. Sjoberg, T. Dyba, and M. Jorgensen, "The future of empirical methods in software engineering research," in *Future of Software Engineering, 2007. FOSE'07*. IEEE, 2007, pp. 358–378.

[120] D. Tomal, *Action research for educators*. Rowman & Littlefield Pub Incorporated, 2010.

[121] S. Jackson, *Research methods and statistics: A critical thinking approach*. Wadsworth Publishing Company, 2011.

[122] A. Pinsonneault and K. Kraemer, "Survey research methodology in management information systems: an assessment," *Journal of Management Information Systems*, vol. 10, no. 2, pp. 75–105, 1993.

[123] P. Biemer, L. Lyberg, and J. Wiley, *Introduction to survey quality*, ser. Wiley Series in Survey Methodology. Wiley, 2003.

[124] S. Presser, *Methods for Testing and Evaluating Survey Questionnaires*, ser. Wiley Series in Survey Methodology. Wiley-Interscience, 2004.

[125] L. Hayduk, *Structural equation modeling with LISREL: Essentials and advances.* Johns Hopkins Univ Pr, 1987.

[126] R. Kline, *Principles and practice of structural equation modeling.* The Guilford Press, 2010.

[127] M. Lovric, *International Encyclopedia of Statistical Science*, 1st ed. Springer, 2011.

[128] G. Hancock, *Structural equation modeling: A second course.* Information Age Pub Inc, 2006, vol. 1.

[129] R. Schumacker and R. Lomax, *A beginner's guide to structural equation modeling.* Lawrence Erlbaum, 2004, vol. 1.

[130] P. Cuttance and R. Ecob, *Structural modeling by example: Applications in educational, sociological, and behavioral research.* Cambridge Univ Pr, 2009.

[131] A. Vieira, *Interactive LISREL in Practice: Getting Started with a SIMPLIS Approach.* Springer Verlag, 2011.

[132] T. Raykov and G. Marcoulides, *A first course in structural equation modeling.* Lawrence Erlbaum, 2006.

[133] P. Barrett, "Structural equation modelling: Adjudging model fit," *Personality and Individual Differences*, vol. 42, no. 5, pp. 815–824, 2007.

[134] G. Cheung and R. Rensvold, "Evaluating goodness-of-fit indexes for testing measurement invariance," *Structural Equation Modeling*, vol. 9, no. 2, pp. 233–255, 2002.

[135] M. Browne and R. Cudeck, "Alternative ways of assessing model fit." *Testing structural equation models (1993) Bollen, Kenneth A.; Long, J. Scott. Newbury Park: Sage Publications.*, 1993.

[136] R. Bagozzi and Y. Yi, "On the evaluation of structural equation models," *Journal of the academy of marketing science*, vol. 16, no. 1, pp. 74–94, 1988.

[137] J. Cote, R. Netemeyer, and P. Bentler, "Improving model fit by correlating errors," *Journal of Consumer Psychology*, vol. 10, no. 1/2, pp. 87–88, 2001.

[138] R. Ping, "On assuring valid measures for theoretical models using survey data," *Journal of Business Research*, vol. 57, no. 2, pp. 125–141, 2004.

[139] J. Kalat, *Introduction to psychology.* Wadsworth Pub Co, 2010.

[140] B. Glaser and A. Strauss, *The discovery of grounded theory: Strategies for qualitative research.* Chicago:Aldine, 1967.

[141] B. Glaser, *Theoretical sensitivity: Advances in the methodology of grounded theory.* Sociology Press, 1978, vol. 2.

[142] J. Corbin and A. Strauss, *Basics of qualitative research: Techniques and procedures for developing grounded theory.* Sage Publications, 2008.

[143] J. Creswell, *Qualitative inquiry and research design: Choosing among five approaches.* Sage Publications, 2012.

[144] A. Bryant and K. Charmaz, *The Sage handbook of grounded theory.* Sage Publications, 2010.

[145] G. Coleman and R. O'Connor, "Using grounded theory to understand software process improvement: A study of irish software product companies," *Information and Software Technology*, vol. 49, no. 6, pp. 654–667, Jun. 2007.

[146] G. Coleman and R. OConnor, "Investigating software process in practice: A grounded theory perspective," *Journal of Systems and Software*, vol. 81, no. 5, pp. 772–784, 2008.

[147] J. Smith, D. Wright, and G. Breakwell, *Research methods in psychology.* SAGE Publications, 2012.

[148] B. Evans, C. Woodall, D. Marks, C. Willig, C. Sykes, and M. Murray, *Health psychology: Theory, research and practice.* Sage Publications, 2005.

[149] D. Smith, Z. Huang, J. Preece, and A. Sears, "The effects of using a triangulation approach of evaluation methodologies to examine the usability of a university website," *Human Computer Interaction Development & Management*, p. 243, 2002.

[150] D. Howitt and D. Cramer, *Introduction to research methods in psychology.* Prentice Hall, 2011.

[151] J. Kontio, L. Lehtola, and J. Bragge, "Using the focus group method in software engineering: obtaining practitioner and user experiences," in *Empirical Software Engineering, 2004. ISESE'04. Proceedings. 2004 International Symposium on.* IEEE, 2004, pp. 271–280.

[152] J. Kontio, J. Bragge, and L. Lehtola, "The focus group method as an empirical tool in software engineering," *Guide to Advanced Empirical Software Engineering*, pp. 93–116, 2008.

[153] K. Salen and E. Zimmerman, *Rules of play: Game design fundamentals.* MIT press, 2003.

[154] E. Adams and J. Dormans, *Game Mechanics: Advanced Game Design.* New Riders, 2012.

[155] C. Abt, *Serious games.* University Press of Amer, 1987.

[156] G. Zichermann and C. Cunningham, *Gamification by Design: Implementing Game Mechanics in Web and Mobile Apps.* O'Reilly Media, 2011.

[157] S. Deterding, R. Khaled, L. Nacke, and D. Dixon, "Gamification: Toward a definition," in *Proceedings of the 2011 Annual Conference Extended*

Abstracts on Human Factors in Computing Systems. ACM, New York, 2011.

[158] D. Flatla, C. Gutwin, L. Nacke, S. Bateman, and R. Mandryk, "Calibration games: making calibration tasks enjoyable by adding motivating game elements," in *Proceedings of the 24th annual ACM symposium on User interface software and technology*. ACM, 2011, pp. 403–412.

[159] S. Deterding, M. Sicart, L. Nacke, K. O'Hara, and D. Dixon, "Gamification. using game-design elements in non-gaming contexts," in *Proceedings of the 2011 annual conference extended abstracts on Human factors in computing systems*. ACM, 2011, pp. 2425–2428.

[160] K. Huotari and J. Hamari, "Gamification from the perspective of service marketing," in *Proc. CHI 2011 Workshop Gamification*, 2011.

[161] S. Deterding, D. Dixon, R. Khaled, and L. Nacke, "From game design elements to gamefulness: defining gamification," in *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments*. New York, NY, USA: ACM, 2011, pp. 9–15.

[162] J. Hamari and V. Eranti, "Framework for designing and evaluating game achievements," *Proc. DiGRA 2011: Think Design Play*, 2011.

[163] F. Groh, "Gamification: State of the art definition and utilization," *Institute of Media Informatics Ulm University*, p. 39, 2012.

[164] J. Von Neumann, O. Morgenstern, A. Rubinstein, and H. Kuhn, *Theory of games and economic behavior*. Princeton Univ Pr, 2007.

[165] D. Schwartz, *Encyclopedia of knowledge management*. IGI Global, 2006.

[166] M. J. Osborne and A. Rubinstein, *A course in game theory*. MIT Press, 1994.

[167] R. Gilles, *The Cooperative Game Theory of Networks and Hierarchies*. Springer Verlag, 2010.

[168] A. K. Dixit and S. Skeath, *Games of Strategy*. WW Norton New York, Jun. 1999.

[169] K. G. Binmore, *Playing for real*. Oxford University Press US, 2007.

[170] B. Lagesse, "A Game-Theoretical model for task assignment in project management," in *2006 IEEE International Conference on Management of Innovation and Technology*, Singapore, 2006, pp. 678–680.

[171] A. Cockburn, "The end of software engineering and the start of economic-cooperative gaming," *COMSIS*, vol. 1, no. 1, pp. 1–32, 2004.

[172] A. Cockburn, *Agile software development: the cooperative game*. Addison-Wesley, 2007.

[173] R. L. Baskerville, L. Levine, B. Ramesh, and J. Pries-Heje, "The high speed balancing game: How software companies cope with internet speed," *Scandinavian Journal of Information Systems*, vol. 16, no. 1, pp. 11–54, 2004.

[174] S.-P. Ko, H.-K. Sung, and K.-W. Lee, "Study to secure reliability of measurement data through application of game theory," in *Proceedings of the 30th EUROMICRO Conference*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 380–386.

[175] R. Holeman, "The software process improvement game," in *Software Engineering Education*, ser. Lecture Notes in Computer Science, R. Ibrahim, Ed. Springer, 1995, vol. 895, pp. 259–261.

[176] P. Ogland, "The game of software process improvement: Some reflections on players, strategies and payoff," *Norsk konferanse for organisasjoners bruk av informasjonsteknologi (NOKOBIT-16)*, pp. 209–223, November 2009.

[177] K. K. Vajja and P. TV, "Quality attribute game: a game theory based techniquefor software architecture design," in *Proceeding of the 2nd an-*

nual conference on India software engineering conference. Pune, India: ACM, 2009, pp. 133–134.

[178] V. Sazawal and N. Sudan, "Modeling software evolution with game theory," *Trustworthy Software Development Processes*, vol. 5543, pp. 354–365, 2009.

[179] G. Bavota, R. Oliveto, A. De Lucia, G. Antoniol, and Y. Gueheneuc, "Playing with refactoring: Identifying extract class opportunities through game theory," in *Software Maintenance (ICSM), 2010 IEEE International Conference on*. IEEE, pp. 1–5.

[180] O. Hazzan and Y. Dubinsky, "Social perspective of software development methods: The case of the prisoner dilemma and extreme programming," in *Extreme Programming and Agile Processes in Software Engineering*. Springer, 2005, pp. 74–81.

[181] L. Feijs, "Prisoner dilemma in software testing," *Computer Science Reports*, vol. 1, pp. 65–80, 2001.

[182] N. V. Oza, "Game theory perspectives on client: vendor relationships in offshore software outsourcing," in *Proceedings of the 2006 international workshop on Economics driven software engineering research*, vol. 27, no. 27. New York, NY, USA: ACM, 2006, pp. 49–54.

[183] M. Klein, G. Moreno, D. Parkes, and K. Wallnau, "Designing for incentives: better information sharing for better software engineering," in *Proceedings of the FSE/SDP workshop on Future of software engineering research*, ser. FoSER '10. ACM, 2010, pp. 195–200.

[184] L. Hurwicz and S. Reiter, *Designing economic mechanisms*. Cambridge Univ. Pr., May 2006.

[185] F. Hayek, "The use of knowledge in society," *American Economic Review*, vol. 35, no. 4, pp. 519–530, 1945.

[186] J. C. Harsanyi, "Games with incomplete information played by "Bayesian" players, I-III: part i. the basic model," *Management Science*, vol. 50, no. 12_supplement, pp. 1804–1817, Dec. 2004.

[187] X. Zhao, F. Fang, and A. Whinston, "An economic mechanism for better internet security," *Decision Support Systems*, vol. 45, no. 4, pp. 811–821, 2008.

[188] T. Stef-Praun and V. Rego, "Ws-auction: Mechanism design for aweb services market," in *Distributed Computing Systems Workshops, 2006. ICDCS Workshops 2006. 26th IEEE International Conference on.* IEEE, 2006, pp. 41–41.

[189] E. Friedman and D. Parkes, "Pricing wifi at starbucks: issues in online mechanism design," in *Proceedings of the 4th ACM conference on Electronic commerce.* ACM, 2003, pp. 240–241.

[190] H. Ziv, D. Richardson, and R. Klosch, "The uncertainty principle in software engineering," in *Proceedings of the 19th International Conference on Software Engineering (ICSE'97)*, 1997.

[191] H. Barki and J. Hartwick, "Interpersonal conflict and its management in information system development," *Mis Quarterly*, p. 195â"228, 2001.

[192] X. Zhang, J. S. Dhaliwal, M. L. Gillenson, and G. Moeller, "Sources of conflict between developers and testers in software development," *AMCIS 2008 Proceedings*, p. 313, 2008.

[193] R. H. Rasch and H. L. Tosi, "Factors affecting software developers' performance: An integrated approach," *MIS Quarterly*, vol. 16, no. 3, pp. 395–413, 1992.

[194] D. Narayan and M. Cassidy, "A dimensional approach to measuring social capital: development and validation of a social capital inventory," *Current Sociology*, vol. 49, no. 2, p. 59, 2001.

[195] N. Madhavji, M. Lehman, D. Perry, and J. Ramil, *Software evolution and feedback*. Wiley Online Library, 2006.

[196] J. Moore, *The death of competition: leadership and strategy in the age of business ecosystems*. HarperBusiness New York, 1996.

[197] E. Mitleton-Kelly, *Complex systems and evolutionary perspectives on organisations*. Emerald Group Publishing, Sep. 2003.

[198] C. Shapiro and H. Varian, *Information rules: a strategic guide to the network economy*. Harvard Business Press, 1999.

[199] S. Shariq, "Sense making and artifacts: an exploration into the role of tools in knowledge management," *Journal of Knowledge Management*, vol. 2, no. 2, pp. 10–19, 1998.

[200] C. Baldwin and K. Clark, *Design rules: The power of modularity*. The MIT Press, 2000.

[201] E. W. Dijkstra, "On the role of scientific thought," in *Selected Writings on Computing: A Personal Perspective*. Springer-Verlag, 1982, pp. 60–66.

[202] M. Cluts, "The evolution of artifacts in cooperative work: constructing meaning through activity," in *Proceedings of the 2003 international ACM SIGGROUP conference on Supporting group work*. ACM, 2003, pp. 144–152.

[203] S. Misterek, K. Dooley, and J. Anderson, "Productivity as a performance measure," *International Journal of Operations & Production Management*, vol. 12, no. 1, pp. 29–45, 1992.

[204] J. Prokopenko, *Productivity management: a practical handbook*. International Labour Office, 1987.

[205] S. Tangen, "Demystifying productivity and performance," *International Journal of Productivity and performance management*, vol. 54, no. 1, pp. 34–46, 2005.

[206] E. Brynjolfsson, "The productivity paradox of information technology," *Communications of the ACM*, vol. 36, no. 12, pp. 66–77, 1993.

[207] D. Sink, T. Tuttle, and S. Shin, *Planning and measurement in your organization of the future.* Industrial engineering and management Press Norcross, Georgia, 1989.

[208] M. Jackson and P. Petersson, "Productivity–an overall measure of competitiveness," in *Proceedings of the Second Workshop on Intelligent Manufacturing Systems, Leuven, Belgium*, 1999, pp. 573–81.

[209] M. Chemuturi, *Software Estimation Best Practices, Tools & Techniques: A Complete Guide for Software Project Estimators.* J. Ross Publishing, 2009.

[210] M. Zelkowitz, *Advances in Computers: Highly Dependable Software.* Gulf Professional Publishing, 2003.

[211] S. Kan, *Metrics and Models in Software Quality Engineering, Second Edition.* Prentice Hall, 2003.

[212] A. Albrecht, "Measuring application development productivity," in *Proceedings of the Joint SHARE/GUIDE/IBM Application Development Symposium*, vol. 10. SHARE Inc. and GUIDE International Corp. Monterey, CA, 1979, pp. 83–92.

[213] K. Maxwell and P. Forselius, "Benchmarking software development productivity," *IEEE Software*, vol. 17, no. 1, pp. 80–88, 2000.

[214] M. Bundschuh and C. Dekkers, *The IT measurement compendium: estimating and benchmarking success with functional size measurement.* Springer, 2008.

[215] S. Tangen, "Understanding the concept of productivity," in *Proceedings of the 7th Asia-Pacific Industrial Engineering and Management Systems Conference, Taipei*, 2002, pp. 18–20.

[216] C. Dale and H. Van Der Zee, "Software productivity metrics: who needs them?" *Information and Software Technology*, vol. 34, no. 11, pp. 731–738, 1992.

[217] A. Trendowicz and J. Munch, "Factors influencing software development productivitystate-of-the-art and industrial experiences," *Advances in computers*, vol. 77, pp. 185–241, 2009.

[218] C. Jones, *Estimating Software Costs: Bringing Realism to Estimating*. McGraw-Hill Companies, 2007.

[219] T. K. Abdel-Hamid, "The slippery path to productivity improvement," *IEEE Software*, vol. 13, no. 4, pp. 43–52, July 1996.

[220] T. Gilb and S. Finzi, *Principles of software engineering management*. Addison-Wesley Wokingham, UK, 1988.

[221] W. Scacchi, "Understanding software productivity," *Software Engineering and Knowledge Engineering: Trends for the Next Decade*, vol. 4, pp. 273–316, 1995.

[222] P. Hantos and M. Gisbert, "Identifying software productivity improvement approaches and risks: construction industry case study," *IEEE Software*, vol. 17, no. 1, p. 56, 2000.

[223] I. D. Steiner, *Group process and productivity*. Academic Press New York, 1972.

[224] T. K. Abdel-Hamid and S. E. Madnick, "Lessons learned from modeling the dynamics of software development," *Communications of the ACM*, vol. 32, no. 12, pp. 1426–1438, Dec. 1989.

[225] B. Boehm, "Improving software productivity," *Computer*, vol. 20, no. 9, pp. 43 –57, sept. 1987.

[226] S. Pfleeger, "Model of software effort and productivity," *Information and Software Technology*, vol. 33, no. 3, pp. 224–231, 1991.

[227] K. Maxwell, *Applied statistics for software managers.* Prentice Hall PTR, 2002.

[228] S. C. d. B. Sampaio, E. A. Barros, G. S. d. Aquino Junior, M. J. C. e. Silva, and S. R. d. L. Meira, "A review of productivity factors and strategies on software development," *Proceedings of the 2010 Fifth International Conference on Software Engineering Advances*, pp. 196–204, 2010.

[229] B. Curtis, H. Krasner, and N. Iscoe, "A field study of the software design process for large systems," *Communications of the ACM*, vol. 31, no. 11, pp. 1268–1287, 1988.

[230] W. Scacchi, "Understanding software productivity: towards a knowledge-based approach," *International Journal of Software Engineering and Knowledge Engineering*, vol. 1, no. 3, pp. 293–321, 1991.

[231] W. Yu, D. Smith, and S. Huang, "Software productivity measurements," in *Computer Software and Applications Conference, 1991. COMPSAC'91., Proceedings of the Fifteenth Annual International.* IEEE, 1991, pp. 558–564.

[232] S. Beecham, N. Baddoo, T. Hall, H. Robinson, and H. Sharp, "Motivation in software engineering: A systematic literature review," *Information and Software Technology*, vol. 50, no. 9-10, pp. 860–878, Aug. 2008.

[233] T. Hall, N. Baddoo, S. Beecham, H. Robinson, and H. Sharp, "A systematic review of theory use in studies investigating the motivations of software engineers," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 18, no. 3, pp. 1–29, 2009.

[234] H. Sharp, N. Baddoo, S. Beecham, T. Hall, and H. Robinson, "Models of motivation in software engineering," *Information and Software Technology*, vol. 51, no. 1, pp. 219–233, 2009.

[235] B. Boehm and P. Papaccio, "Understanding and controlling software costs," *IEEE Transactions on Software Engineering*, vol. 14, no. 10, pp. 1462–1477, 1988.

[236] S. Faraj and L. Sproull, "Coordinating expertise in software development teams," *Management Science*, vol. 46, no. 12, pp. 1554–1568, 2000.

[237] V. Basili, L. Briand, and W. Melo, "How reuse influences productivity in object-oriented systems," *Communications of the ACM*, vol. 39, no. 10, pp. 104–116, 1996.

[238] B. Boehm, "Managing software productivity and reuse," *Computer*, vol. 32, no. 9, pp. 111–113, 2002.

[239] D. Simmons, *Software Organization Productivity.* Software Productivity Improvement Laboratory,Department of Computer Science Texas A & M University, 2007.

[240] Z. Jiang, P. Naudé, and C. Comstock, "An investigation on the variation of software development productivity," *International Journal of Computer, Information, and Systems Sciences, and Engineering*, vol. 1, no. 2, pp. 72–81, 2007.

[241] Z. Jiang and C. Comstock, "The factors significant to software development productivity," in *Proceedings of World Academy of Science, Engineering and Technology*, vol. 21. Citeseer, 2007, pp. 160–164.

[242] I. Chiang and V. Mookerjee, "Improving software team productivity," *Communications of the ACM*, vol. 47, no. 5, pp. 89–93, 2004.

[243] C. Behrens, "Measuring the productivity of computer systems development activities with function points," *IEEE Transactions on Software Engineering*, vol. SE-9, no. 6, pp. 648–652, 1983.

[244] S. Wagner and M. Ruhe, "A structured review of productivity factors in software development," *Institut fur Informatik-Technische Universitat Munchen, Tech. Rep. Technical Report TUMI0832*, 2008.

[245] M. Yilmaz and R. V. O'Connor, "Social capital as a determinant factor of software development productivity: An empirical study using structural equation modeling," *International Journal of Human Capital and Information Technology Professionals (IJHCITP)*, vol. 3, no. 2, pp. 40–62, 2012.

[246] B. Boehm and K. J. Sullivan, "Software economics: a roadmap," in *Proceedings of the Conference on The Future of Software Engineering*. Limerick, Ireland: ACM, 2000, pp. 319–343.

[247] W. Welfe, *Knowledge-based economies: models and methods*. Peter Lang Publishing, 2009.

[248] K. Marx, *Capital: a critical analysis of capitalist production*. Appleton, 1889.

[249] N. Lin, *Social capital: A theory of social structure and action*. Cambridge Univ Press, 2002.

[250] P. Bordieu, "The forms of capital," *Handbook of Theory and Research for the Sociology of Education*, vol. 241, pp. 241–258, 1986.

[251] A. Portes, "Social capital: its origins and applications in modern sociology," *Annual review of sociology*, vol. 24, no. 1, pp. 1–24, 1998.

[252] F. Fukuyama, *Trust: The social virtues and the creation of prosperity*. Free Pr, 1996.

[253] N. Christakis and J. Fowler, *Connected: The surprising power of our social networks and how they shape our lives*. Little, Brown and Company, 2009.

[254] J. Coleman, *Foundations of social theory*. Belknap Press, 1994.

[255] J. Coleman, "Social capital in the creation of human capital," *The American Journal of Sociology*, vol. 94, no. 1, pp. 95–120, 1988.

[256] G. Homans, "Social behavior as exchange," *American journal of sociology*, vol. 63, no. 6, pp. 597–606, 1958.

[257] P. Bourdieu and L. Wacquant, *An invitation to reflexive sociology*. University of Chicago Press, 1992.

[258] L. Barnett, "Social Productivity, Law, and the Regulation of Conflicts of Interest in the Investment Industry," *Cardozo Public Law, Policy and Ethics Journal*, vol. 3, p. 793, 2004.

[259] T. Stober and U. Hansmann, *Agile Software Development: Best Practices for Large Software Development Projects*. Springer-Verlag, 2009.

[260] S. Koh and S. Maguire, *Information and Communication Technologies Management in Turbulent Business Environments*, ser. Premier Reference Source. Information Science Reference, 2009.

[261] O. Hazzan and Y. Dubinsky, *Agile Software Engineering*, ser. Undergraduate Topics in Computer Science. Springer, 2008.

[262] D. Anderson, *Agile management for software engineering: applying the theory of constraints for business results*, ser. The Coad series. Prentice Hall PTR, 2004.

[263] A. Kelly, *Changing software development: Learning to become agile*. Wiley, 2008.

[264] D. Churchville, *Agile Thinking: Leading successful software projects and teams*. ExtremePlanner Software, Dec. 2008.

[265] K. Krippendorff, *Content analysis: An introduction to its methodology*. Sage Publications, 2004.

[266] B. Glaser and A. Strauss, *The discovery of grounded theory: Strategies for qualitative research*. Aldine Transaction, 2007.

[267] E. Raymond, "The cathedral and the bazaar," *Knowledge, Technology & Policy*, vol. 12, no. 3, pp. 23–49, 1999.

[268] S. Sheard, "The value of Twelve systems engineering roles," in *Proceedings of INCOSE*. Citeseer, 1996.

[269] S. Sheard, "Twelve systems engineering roles," in *Proceedings of INCOSE*. Citeseer, 1996.

[270] P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta, *Agile software development methods: Review and Analysis*. Technical Research Centre of Finland, 2002, vol. VTT Publications 478, no. 3.

[271] B. Curtis, W. E. Hefley, and S. A. Miller, *The People Capability Maturity Model(R): Guidelines for Improving the Workforce*. Addison-Wesley Professional, Dec. 2001.

[272] B. Curtis, W. Hefley, and S. Miller, *People CMM: A Framework for Human Capital Management*. Addison-Wesley Professional, 2009.

[273] A. P. Association, *Diagnostic and Statistical Manual of Mental Disorders DSM-IV-TR Fourth Edition*, 4th ed. Amer Psychiatric Pub, Jun. 2000.

[274] C. Jung, H. Baynes, and R. Hull, *Psychological types*. Routledge, 1991.

[275] I. Myers, M. McCaulley, and R. Most, *Manual: A guide to the development and use of the Myers-Briggs Type Indicator*, 1985.

[276] I. Myers, M. McCaulley, N. Quenk, and A. Hammer, *MBTI manual*. Consulting Psychologists Press, 1999.

[277] D. Pittenger, "Measuring the mbti and coming up short," *Journal of Career Planning and Employment*, vol. 54, no. 1, pp. 48–52, 1993.

[278] N. Quenk, *Essentials of Myers-Briggs type indicator assessment*. Wiley, 2009, vol. 66.

[279] D. Wilde, *Teamology: the construction and organization of effective teams*. Springer Verlag, 2008, vol. 10.

[280] S. Cruz, F. da Silva, C. Monteiro, P. Santos, and I. Rossilei, "Personality in software engineering: Preliminary findings from a systematic literature review," in *Evaluation & Assessment in Software Engineering (EASE 2011), 15th Annual Conference on*. IET, 2011, pp. 1–10.

[281] D. Keirsey and M. Bates, *Please understand me: Character & temperament types*. Prometheus Nemesis Michigan, 1984.

[282] K. White, "A preliminary investigation of information systems team structures," *Information & Management*, vol. 7, no. 6, pp. 331–335, 1984.

[283] K. Kaiser and R. Bostrom, "Personality characteristics of mis project teams: An empirical study and action-research design," *MIS Quarterly*, vol. 6, no. 4, pp. 43–60, 1982.

[284] R. Rutherfoord, "Using personality inventories to help form teams for software engineering class projects," *ACM SIGCSE Bulletin*, vol. 33, no. 3, pp. 73–76, 2001.

[285] J. Karn and T. Cowling, "A follow up study of the effect of personality on the performance of software engineering teams," in *Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering*. ACM, 2006, pp. 232–241.

[286] P. Sfetsos, I. Stamelos, L. Angelis, and I. Deligiannis, "An experimental investigation of personality types impact on pair effectiveness in pair programming," *Empirical Software Engineering*, vol. 14, no. 2, pp. 187–226, 2009.

[287] A. Dick and B. Zarnett, "Paired programming and personality traits," *XP2002, Italy*, 2002.

[288] J. Karn, S. Syed-Abdullah, A. Cowling, and M. Holcombe, "A study into the effects of personality type and methodology on cohesion in software engineering teams," *Behaviour & Information Technology*, vol. 26, no. 2, pp. 99–111, 2007.

[289] C. Bush and L. Schkade, "In search of the perfect programmer." *Datamation*, vol. 31, no. 6, pp. 128–132, 1985.

[290] E. Buie, "Psychological type and job satisfaction in scientific computer professionals," *Journal of Psychological Type*, vol. 15, pp. 50–53, 1988.

[291] D. Smith, "The personality of the systems analyst: an investigation," *ACM SIGCPR Computer Personnel*, vol. 12, no. 2, pp. 12–14, 1989.

[292] R. Turley and J. Bieman, "Competencies of exceptional and nonexceptional software engineers," *Journal of Systems and Software*, vol. 28, no. 1, pp. 19–38, 1995.

[293] L. Hardiman, "Personality types and software engineers," *Computer*, vol. 30, no. 10, pp. 10–10, 1997.

[294] L. Capretz, "Personality types in software engineering," *International Journal of Human-Computer Studies*, vol. 58, no. 2, pp. 207–214, 2003.

[295] R. Sach, M. Petre, and H. Sharp, "The use of mbti in software engineering," in *22nd Annual Psychology of Programming Interest Group 2010*, September 2010.

[296] A. Da Cunha and D. Greathead, "Does personality matter?: an analysis of code-review ability," *Communications of the ACM*, vol. 50, no. 5, pp. 109–112, 2007.

[297] N. Gorla and Y. Lam, "Who should work with whom?: building effective software project teams," *Communications of the ACM*, vol. 47, no. 6, pp. 79–82, 2004.

[298] D. Varona, L. Capretz, and Y. Piñero, "Personality types of cuban software developers," *Global Journal of Engineering Education*, vol. 13, no. 2, 2011.

[299] D. Varona, L. F. Capretz, Y. Piñero, and A. Raza, "Evolution of software engineers' personality profile," *ACM SIGSOFT Software Engineering Notes*, vol. 37, no. 1, pp. 1–5, Jan. 2012.

[300] L. Capretz, "Software development and personality traits," 2012, invited talk, Tenth International Conference on Computer Applications (ICCA 2012), the University of Computer Studies, in Yangon Myanmar.

[301] L. Capretz and F. Ahmed, "Why do we need personality diversity in software engineering?" *ACM SIGSOFT Software Engineering Notes*, vol. 35, no. 2, pp. 1–11, 2010.

[302] F. Ahmed, L. Capretz, and P. Campbell, "Evaluating the demand for soft skills in software development," *IT Professional*, vol. 14, no. 1, pp. 44–49, 2012.

[303] J. Karn and A. Cowling, "Using ethnographic methods to carry out human factors research in software engineering," *Behavior research methods*, vol. 38, no. 3, pp. 495–503, 2006.

[304] H. Goldsmith, A. Buss, R. Plomin, M. Rothbart, A. Thomas, S. Chess, R. Hinde, and R. McCall, "Roundtable: What is temperament? four approaches," *Child development*, pp. 505–529, 1987.

[305] G. Allport, "Pattern and growth in personality." 1961.

[306] D. Joyce, *Essentials of temperament assessment.* Wiley, 2010, vol. 71.

[307] D. DeCarlo, *eXtreme project management: Using leadership, principles, and tools to deliver value in the face of volatility.* Jossey-Bass, 2004.

[308] E. Scerri, *The periodic table: its story and its significance.* Oxford University Press, USA, 2006.

[309] D. Bradbary and D. Garrett, *Herding chickens: innovative techniques for project management.* Jossey-Bass, 2005.

[310] R. Krueger and M. Casey, *Focus groups: A practical guide for applied research.* Sage Publications, 2009.

[311] K. Joreskog and D. Sorbom, *LISREL 8: users reference guide. Lincolnwood, IL: Scientific Software International.* Inc, 2001.

[312] L. Cronbach, "Coefficient alpha and the internal structure of tests," *Psychometrika*, vol. 16, no. 3, pp. 297–334, 1951.

[313] A. Lehman, *JMP for basic univariate and multivariate statistics: a step-by-step guide*. SAS Publishing, 2005.

[314] P. Kline, *A handbook of test construction: Introduction to psychometric design*. Methuen, 1986.

[315] A. Field, *Discovering statistics using SPSS*. SAGE publications, 2009.

[316] D. Rodríguez, M. Sicilia, E. García, and R. Harrison, "Empirical findings on team size and productivity in software development," *Journal of Systems and Software*, 2011.

[317] F. P. Brooks, *The Mythical Man-Month: Essays on Software Engineering, Anniversary Edition (2nd Edition)*. Addison-Wesley Professional, 1995.

[318] T. Abdel-Hamid and S. E. Madnick, *Software project dynamics: an integrated approach*. Prentice Hall Englewood Cliffs, NJ, 1991.

[319] L. Putnam, "A general empirical solution to the macro software sizing and estimating problem," *IEEE Transactions on Software Engineering*, no. 4, pp. 345–361, 1978.

[320] R. Daft and D. Marcic, *Understanding management*. South-Western Pub, 2006.

[321] M. Hoegl, "Smaller teams–better teamwork: How to keep project teams small," *Business Horizons*, vol. 48, no. 3, pp. 209–214, 2005.

[322] M. Hansen, W. Hurwitz, L. Pritzker, and U. S. B. of the Census, *The estimation and interpretation of gross differences and the simple response variance*. Bureau of the Census, 1963.

[323] J. Cohen, "A coefficient of agreement for nominal scales," *Educational and psychological measurement*, vol. 20, no. 1, pp. 37–46, 1960.

[324] R. Basu, *Implementing quality: a practical guide to tools and techniques: enabling the power of operational excellence.* Cengage Learning Business Press, 2004.

[325] R. Harris, *Information graphics: A comprehensive illustrated reference.* Oxford University Press, USA, 2000.

[326] N. Moe, T. Dingsøyr, and E. Røyrvik, "Putting agile teamwork to the test–an preliminary instrument for empirically assessing and improving agile software development," *Agile Processes in Software Engineering and Extreme Programming*, pp. 114–123, 2009.

[327] M. Ringstad, T. Dingsøyr, and N. Brede Moe, "Agile process improvement: Diagnosis and planning to improve teamwork," *Systems, Software and Service Process Improvement*, vol. 172, no. 1, pp. 167–178, 2011.

[328] W. Humphrey, *TSP: Leading a Development Team.* Addison-Wesley Professional, 2005.

[329] L. Gottschalk, *Content analysis of verbal behavior: New findings and clinical applications.* Lawrence Erlbaum Associates, Inc., 1995.

[330] M. Yilmaz and R. O'Connor, "An empirical investigation into social productivity of a software process: An approach by using the structural equation modeling," in *Proceedings of the 18th European System and Software Process Improvement and Innovation Conference (EuroSPI 2011)*, vol. 172. Springer Berlin Heidelberg, 2011, pp. 155–166.

[331] L. Foulds, M. Quaddus, and M. West, "Structural equation modelling of large-scale information system application development productivity: the Hong Kong experience," in *Computer and Information Science, 2007. ICIS 2007. 6th IEEE/ACIS International Conference on.* IEEE, 2007, pp. 724–731.

[332] S. Ng, Y. Wong, and J. Wong, "A structural equation model of feasibility evaluation and project success for public–private partnerships in hong

kong," *Engineering Management, IEEE Transactions on*, vol. 57, no. 2, pp. 310–322, 2010.

[333] L. Capretz and F. Ahmed, "Making sense of software development and personality types," *IT Professional*, vol. 12, no. 1, pp. 6–13, 2010.

[334] O. Mazni, S. Syed-Abdullah, and N. Hussin, "Analyzing personality types to predict team performance," in *Science and Social Research (CSSR), 2010 International Conference on.* IEEE, 2010, pp. 624–628.

[335] T. Lewis and W. Smith, "Building software engineering teams that work: The impact of dominance on group conflict and performance outcomes," in *Frontiers in Education Conference, 2008. FIE 2008. 38th Annual.* IEEE, 2008, pp. S3H–1.

[336] Z. Su-li and X. Ke-fan, "Research on entrepreneurial team members' personality traits influence on group risk decision-making," in *Management Science and Engineering (ICMSE), 2010 International Conference on.* IEEE, 2010, pp. 937–942.

[337] M. Conway, "How do committees invent," *Datamation*, vol. 14, no. 4, pp. 28–31, 1968.

[338] M. Rothstein and R. Goffin, "The use of personality measures in personnel selection: What does current research support?" *Human Resource Management Review*, vol. 16, no. 2, pp. 155–180, 2006.

Part VI

Appendices

Appendix A

Survey Instrument

A Survey for identifying the factors affecting productivity

May I firstly introduce myself: My name is Murat Yilmaz. I am a doctoral researcher at Lero - The Irish Software Engineering Research Center at Dublin City University, Ireland. Over the last two years, as a part of my PhD project, I am working on identifying several critical factors that are affecting the productivity of software development organizations.

To achieve precise results with this research, your help would be greatly appreciated for analyzing the identified factors in detail via our questionnaire below. The same questionnaire will be used for all participants, which will be treated in strict confidentiality, therefore there is no requirement for names or other personal details. Your responses will not be shared with third parties.Please take a few minutes to fill out our survey. It will 30 minutes to complate. Please carefully read the following statements. Indicate your agreement or lack of agreement with each of the statements. Should you need any further information, please do not hesitate to contact me at murat.yilmaz@computing.dcu.ie, and visit my web page:
http://www.computing.dcu.ie/~myilmaz/academic/index.html.

Please state your gender:

- ☐ Male
- ☐ Female

Since graduating, how many years how many years have you spent working in the Software Development Industry.

.....

Please state your current role/job title (e.g. software developer, software tester, etc.):

.....

How long have you been served in your current position?

.....

Software Productivity Factors of Software Development

Software development productivity is a success measure of software development. It is a ratio between the functional value of software artifacts that are produced to the workforce and costs of producing it. The goal of this survey is to understand your opinion on several factors that may affect productivity of software development.

Motivation

1. The level of an individual's motivation has a significant affect on the productivity of software development.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
☐	☐	☐	☐	☐

2. The level of interest that people have for their assigned tasks directly affects the productivity of software development.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
☐	☐	☐	☐	☐

Management Quality, e.g. process, development tools, programming languages

3. The productivity of software development is affected by the choice of development process or methodology (e.g. waterfall, iterative, agile, etc.).

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
☐	☐	☐	☐	☐

4. The choice of programming language has a significant impact on software development productivity.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
☐	☐	☐	☐	☐

5. The tools and technologies used for software development have a major effect on software development productivity.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
☐	☐	☐	☐	☐

Complexity Issues, e.g. task, process, product

6. Working on complex and challenging tasks as opposed to routine tasks will improve software development productivity.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
☐	☐	☐	☐	☐

7. In a development of large, complex structured programming projects, it is more difficult to get an accurate assessment of software development productivity.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
☐	☐	☐	☐	☐

8. The number of tasks that are identified in a software project and their complex connections has a major impact on software development productivity.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
☐	☐	☐	☐	☐

Work Environment

9. The physical layout, furnishings and office support services are important project resources that affect software development productivity.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
☐	☐	☐	☐	☐

Re-usability

10. As opposed to writing every line of code starting from scratch, it would be better for a software project to use some off-the-self product or a library to improve software development productivity.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
☐	☐	☐	☐	☐

Requirements Stability

11. The ability of an organization to stabilize customers' requirements (i.e. expectations) has a significant affect on productivity of software development.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
☐	☐	☐	☐	☐

12. The changes in requirements of a project could have a significant impact on software development productivity.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
☐	☐	☐	☐	☐

Team Issues, e.g. size, organization, location

13. Software development productivity is affected by team size.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
☐	☐	☐	☐	☐

14. In order to improve software development productivity, the team members must frequently communicate verbally with each other.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
☐	☐	☐	☐	☐

15. From a software development productivity perspective, it is important that a team member should have an ability to interpret non-verbal communications such as facial expression, eye contact, etc.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
☐	☐	☐	☐	☐

16. Software development productivity is not be affected by having team members in different physical locations.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
☐	☐	☐	☐	☐

17. An ideal software team should be a self-sufficient group which means its members are able to solve their problems that occur during development.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
☐	☐	☐	☐	☐

18. How many members are in your immediate development team?

☐☐☐ Members

19. In your view, what proportion of your team members are operating at high levels of productivity?

☐☐☐ Members

Social Productivity Factors

Social productivity involves targeting the quality of social interactions in order to bring about productivity improvements. Furthermore, by designing a better communication and social structure (i.e. mechanism) for software organization, we aim to improve the social structure and welfare of a software organization so as to improve the organizational outputs, i.e. Software production.

Team Leader, e.g. conflicts, reputation

20. A conflict between two members of a team should be addressed by a team leader.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
☐	☐	☐	☐	☐

21. To improve team performance, a team leader's skills are an important factor.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
☐	☐	☐	☐	☐

Social Interaction, life, communication

22. A member of a team should communicate with every other member of a team.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
☐	☐	☐	☐	☐

23. I would like to have a social life with my teammates outside of the workplace.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
☐	☐	☐	☐	☐

Information Awareness

24. In a well functioning team setting, I should know what everyone is doing, and everyone should know what I am doing.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
☐	☐	☐	☐	☐

25. A collective team memory involves awareness of what actions will have a potential impact on goals and objectives of a team as a whole.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
☐	☐	☐	☐	☐

Team Cohesion

26. Individual team members should be united in the service of the team goals.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
☐	☐	☐	☐	☐

27. A team's ability to work together depends on how its members enjoy each others company.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
☐	☐	☐	☐	☐

Fairness

28. I lose all sense of fairness, if I see some other members of my team are doing less work than me.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
☐	☐	☐	☐	☐

29. I am confident that my teammates are likely to be successful to achieve their tasks when they are assigned a fair allocation of the work.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
☐	☐	☐	☐	☐

Frequent Meetings

30. A team should meet on very regular basis to be informed about each others progress.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
☐	☐	☐	☐	☐

Social Trust

31. I am aware of my reliance on my teammates.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
☐	☐	☐	☐	☐

Social Capital Factors

Social capital is an important network-based element that provides insight to individuals for maximizing their productivity by valuing tangible resources in a social setting such as software development. Therefore, we argue that it is vitally important to identify and correlate social determinants that are potentially affecting the software development productivity.

Neighborhood Connections

32. I think the people around me (i.e. my social connections) make a significant impact on my career.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
☐	☐	☐	☐	☐

33. In the past, I have secured new jobs by using my network of social connections.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
☐	☐	☐	☐	☐

Group Characteristics

34. The people that I socially connected to have a significant impact on my career success.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
☐	☐	☐	☐	☐

35. Gathering different personality types of people together is essential to accomplish team objectives.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
☐	☐	☐	☐	☐

Generalized Norms

36. I can improve the value of my social relationships with colleagues, if I behave in a manner that respects the accepted social norms.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
☐	☐	☐	☐	☐

Togetherness

37. I add people to my social circles (networks) regarding to their potential benefits to me.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
☐	☐	☐	☐	☐

Everyday sociability

38. I think people with extra social skills are more collaborative than the ones with less social skills.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
☐	☐	☐	☐	☐

Volunteerism

39. I prefer to volunteer for extra work so as to extend my network of connections.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
☐	☐	☐	☐	☐

40. Volunteering can help individuals to better understand different social types of people.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
☐	☐	☐	☐	☐

Trust

41. The ability of people to trust each other and maintain cooperative relationships is the result of their social experiences.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
☐	☐	☐	☐	☐

Complementary Questions

42. A proper alignment between the goals of individuals and organizational objectives has a significant impact on the productivity of software development.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
☐	☐	☐	☐	☐

43. While setting up software teams, it is important to consider a method based on individuals personality types rather than using an ad-hoc method.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
☐	☐	☐	☐	☐

44. Revealing the position of a team member in a social structure together with investigation of his or her personality type, will help us to improve the software development productivity.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
☐	☐	☐	☐	☐

45. Improving the social and communication structure of a software organization will incorporate features like effective team formation.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
☐	☐	☐	☐	☐

Personality type questions

This part aims to indicate the importance of the factors potentially affecting an individual's personality characteristics. Please select the degree of importance of the factors for each question.

46. The ability to engage in social interactions.

Very Important	Important	Moderately Important	Little Importance
☐	☐	☐	☐

47. The capacity to act in the pursuit of socially valued goals (i.e. social courage).

Very Important	Important	Moderately Important	Little Importance
☐	☐	☐	☐

48. The skills and confidence in conversations.

Very Important	Important	Moderately Important	Little Importance
☐	☐	☐	☐

49. The grouping of people as being factual or fictional.

Very Important	Important	Moderately Important	Little Importance
☐	☐	☐	☐

50. The preference of an individual to rely on experiences or hunches.

Very Important	Important	Moderately Important	Little Importance
☐	☐	☐	☐

51. The preference for focusing on specification or generalizations.

Very Important	Important	Moderately Important	Little Importance
☐	☐	☐	☐

52. An individual's ability for being firm or gentle.

Very Important	Important	Moderately Important	Little Importance
☐	☐	☐	☐

53. Maintaining a focus on individual's values or following general principles.

Very Important	Important	Moderately Important	Little Importance
☐	☐	☐	☐

54. The degree of valuing thoughts over emotions.

Very Important	Important	Moderately Important	Little Importance
□	□	□	□

55. Developing skills for either planning ahead or adapt as you go.

Very Important	Important	Moderately Important	Little Importance
□	□	□	□

56. The valuation of process over product.

Very Important	Important	Moderately Important	Little Importance
□	□	□	□

57. The ability to act quickly and decide fast.

Very Important	Important	Moderately Important	Little Importance
□	□	□	□

Appendix B

Sample Coding for Extroversion

Open Coding	Axial Coding	Selective Coding	Core Category
• A lack of ability to initiate *conversation or* able to maintain a *conversation* with their peers. • Initiating *conversations* to form *social interactions.* • Passive/defensive *interaction* styles tend to withdraw from *conversations* as well as *social interactions*. • Positive *social interactions* affect software team effectiveness. • *Social interactions* cause complex *interrelationships*. • Social *interrelationships* can be complex and difficult to understand	*Conversation* *Interaction* *Interrelationships*	**Social Interactions**	***Extroversion***
• Helping a teammate to develop the *social courage he possess* • A social team may risk arguing in order to improve team productivity • Disputes are caused by *disagreement and may* promote *social courage*	*A Teammate* *A Social Team* *Arguing*	**Social Courage**	
• *Conversationalist loves to be in contact, they practice their skills in night-outs,* parties, and lunches or even with informal chats in the company corridors. • Developing confidence in social activities requires basic *skills* in *conversation especially in daily kick-off meetings* • *Meeting room is like a temple; a confident place to have a chat, our* burn down charts and story cards all over its walls.	*A night out* *Party* *Contact* *Meeting Room*	**Skills and Confidence in Conversations**	

Appendix C

Survey Data

ID NO	SEX	JOB TITLE	WORK _YEAR	WORK _THIS _FIRM	TEAM_ SIZE	IDEAL_ TEAM_ SIZE	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	X14	X15	X16	X17	X20	X21	X22	X23	X24	X25	X26	X27	X28	X29	X30	X31	X32	X33	X34	X35	X36	X37	X38	X39	X40	X41
1	Male	Project manager	10	3	6	4	5	5	4	4	4	4	4	4	4	5	5	3	4	5	3	2	4	4	4	4	3	4	4	4	4	4	4	3	4	3	4	3	3	3	1	1	2	4	4
2	Male	Software developer	5	4	3	1	5	5	3	4	5	5	4	2	5	4	5	4	4	4	2	1	4	4	5	5	2	4	4	3	5	5	5	2	4	2	3	2	4	3	3	2	3	4	4
3	Male	Software developer	1	1	6	4	5	5	5	5	5	5	4	4	5	4	4	4	3	5	4	5	5	4	5	5	4	5	4	5	2	2	4	4	4	4	4	3	4	4	2	4	4	4	4
4	Male	Software architecture	7	2	5	3	5	5	3	4	4	4	3	3	4	4	5	3	3	4	3	3	4	4	5	5	3	4	4	4	5	2	5	4	5	3	2	3	4	3	2	4	5	5	3
5	Male	Software developer	6	6	30	15	5	5	5	5	5	5	5	5	4	5	5	5	5	5	5	2	5	5	5	5	3	5	5	5	5	5	3	5	5	5	3	3	3	5	2	3	5	5
6	Male	Software developer	1	1	4	4	4	5	5	4	4	3	2	5	3	3	3	4	2	5	3	2	5	4	4	5	4	5	3	4	5	4	5	5	5	4	4	3	3	2	2	5	2	3	4
7	Male	Software developer	3	0	5	5	5	5	3	3	3	4	4	3	2	4	2	4	4	4	3	4	4	3	3	4	3	3	4	3	3	3	3	3	3	3	3	3	2	3	3	2	3	3	3
8	Male	Software developer	18	8	5	3	5	5	5	4	4	1	2	3	4	4	4	5	4	3	4	4	3	5	5	5	4	5	4	5	3	4	3	4	4	4	5	4	4	3	3	2	3	4	4
9	Male	Team Leader	13	3	5	5	5	5	4	4	4	4	3	4	5	4	5	4	4	5	2	1	4	5	4	5	3	5	4	5	5	2	4	4	5	2	3	2	3	4	2	2	2	4	2
10	Male	Team Leader	8	1	5	3	5	5	5	5	5	5	4	3	4	4	5	2	4	5	5	2	5	4	4	5	5	4	3	4	4	2	4	4	4	4	4	4	2	1	3	3	4	4	4
11	Male	Software developer	8	2	12	7	4	5	3	3	3	4	3	3	3	2	4	3	2	4	3	2	4	3	5	3	4	4	4	4	2	2	4	3	4	3	3	2	3	2	1	4	3	3	4
12	Male	Software developer	5	4	8	4	5	5	4	5	4	5	4	4	4	5	4	2	2	4	2	2	4	2	4	4	5	4	4	4	4	2	4	4	4	2	4	2	4	2	1	2	4	4	4
13	Male	Software Tester	11	3	11	4	4	4	3	3	4	2	4	4	3	3	4	4	4	4	4	2	4	4	4	4	2	4	4	4	4	4	3	4	4	3	3	3	3	3	2	4	3	3	3
14	Male	Software developer	6	6	2	2	4	4	5	5	4	5	4	4	4	4	4	2	4	5	2	2	5	4	4	4	4	4	4	4	5	4	5	4	4	4	4	3	3	3	2	4	3	4	4
15	Male	Software developer	9	4	9	8	4	5	4	4	4	5	3	4	4	5	4	4	4	3	2	4	4	4	3	2	3	4	4	2	4	4	4	3	3	3	2	4	4	3	3	2	4	3	4
16	Male	Software developer	5	2	5	3	4	5	4	5	4	4	3	4	5	4	4	4	4	5	4	2	4	3	4	5	4	4	4	4	4	4	5	4	5	4	4	4	4	4	3	4	4	4	4
17	Male	Software developer	8	4	8	7	4	4	4	2	4	3	3	2	2	4	4	3	4	4	2	4	4	4	4	4	4	3	3	4	2	2	2	2	2	2	2	3	3	3	3	3	3	3	3
18	Male	Software developer	1	1	6	6	4	4	4	4	5	5	5	4	4	4	4	5	4	4	4	4	2	4	5	4	4	4	3	5	4	4	5	3	4	4	4	5	5	5	4	4	3	4	4
19	Female	Software developer	2	2	6	2	5	5	2	4	5	5	2	5	5	2	5	5	4	4	1	4	4	4	5	5	3	4	4	4	4	5	4	4	4	3	2	2	3	3	2	2	4	4	4
20	Male	Software developer	7	7	5	4	5	5	3	4	4	4	4	4	5	4	4	4	4	2	2	4	4	5	4	4	4	4	4	4	4	4	4	4	4	3	4	3	3	3	1	3	2	3	2
21	Male	Team Leader	14	3	6	3	5	5	3	4	4	3	2	4	5	4	5	5	4	5	4	1	5	4	4	4	3	4	4	4	2	2	4	5	5	5	2	4	4	4	2	1	4	4	3
22	Male	Software architecture	25	4	13	6	4	4	5	5	4	4	3	3	5	4	4	5	3	5	2	3	4	4	5	5	4	4	4	4	3	4	4	5	4	4	3	3	3	3	3	3	3	3	3
23	Male	Software architecture	7	2	3	3	5	4	4	4	4	3	4	3	3	4	4	3	3	4	5	2	1	5	3	4	5	4	4	5	3	4	4	4	3	5	4	4	4	4	3	3	2	2	2	3	4
24	Male	Software developer	2	3	7	4	5	5	5	5	5	2	4	2	4	5	5	5	4	5	2	3	5	4	5	5	4	5	4	5	4	5	2	2	5	5	5	4	5	2	3	2	1	4	4	5	5
25	Male	Software specialist	1	1	5	3	5	5	4	5	5	5	3	5	4	5	5	5	2	5	1	2	4	3	3	4	4	5	4	4	4	3	3	5	4	2	2	4	1	2	3
26	Male	Software developer	5	5	5	5	5	4	3	2	4	5	4	4	5	3	3	4	2	5	3	1	4	4	4	5	5	4	4	4	4	5	2	4	4	5	4	4	4	1	3	2	4	3	3	3
27	Male	Software specialist	1	1	4	4	5	4	4	3	5	5	4	4	3	4	4	1	4	5	2	4	2	1	4	4	5	5	4	4	3	2	2	4	2	4	3	4	5	4	4	2	3	1	1	2	2	3	4
28	Male	Software developer	4	1	15	8	5	4	4	4	5	3	2	3	4	2	2	2	4	3	2	4	5	1	4	4	3	2	3	4	1	4	2	1	4	2	3	2	3	2	1	3	4	3	3
29	Male	Software developer	2	1	5	3	5	4	3	2	4	4	4	5	3	3	3	4	4	3	5	2	4	4	5	2	4	2	3	4	2	2	4	4	4	3	2	3	3	2	2	4	2	3	4
30	Male	Software developer	1	1	6	6	5	5	4	4	5	3	3	4	4	3	4	3	4	3	4	5	4	2	4	4	4	5	4	4	4	4	4	4	4	4	4	3	2	4	4	3	3	4	4
31	Male	Software Tester	8	2	6	7	5	5	4	3	4	4	3	3	2	5	4	4	4	4	3	3	2	2	4	4	4	3	4	4	4	4	5	4	4	4	5	4	3	3	4	3	2	3	2	3	3
32	Male	Software specialist	1	1	5	5	5	5	5	5	5	4	5	5	5	4	5	4	4	4	3	2	2	4	5	5	4	4	4	5	5	5	5	4	5	4	4	5	4	4	4	3	2	3	2	4	4
33	Male	Software specialist	2	1	4	4	5	5	4	4	4	4	3	4	4	4	4	4	4	5	3	2	4	4	4	4	4	5	3	4	4	3	4	5	5	4	4	4	3	2	3	2	4	4	4
34	Male	Software developer	14	2	1	6	5	4	4	4	4	3	3	4	3	3	4	5	5	4	4	1	4	5	4	4	4	4	4	4	4	3	4	4	4	2	4	3	5	4	2	4	1	3	4
35	Female	Project manager	20	12	5	3	5	5	2	2	5	4	5	5	5	5	5	4	2	5	1	1	5	1	4	5	2	5	5	3	4	5	4	5	5	4	5	5	2	3	2	1	5	5	5
36	Female	IT Specialist	8	2	7	1	5	5	3	2	4	4	4	3	4	5	5	3	3	5	2	2	4	4	5	5	3	4	4	4	4	5	4	5	5	4	4	4	3	4	2	2	2	3	4
37	Female	IT Specialist	12	2	5	5	5	4	4	5	4	4	3	4	5	3	4	2	2	4	4	2	5	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	2	2	3	4	4
38	Male	Software developer	7	7	5	5	5	5	4	4	4	4	2	2	4	3	4	3	4	5	4	1	4	4	5	5	4	5	5	5	3	4	4	4	4	3	4	3	3	4	2	4	2	2	4
39	Male	Project manager	11	1	6	4	5	5	4	2	5	3	2	4	4	3	4	4	2	4	4	2	4	2	4	4	3	4	3	4	4	4	4	3	4	2	4	3	4	2	1	1	1	1	3
40	Female	Software specialist	1	1	6	6	5	4	4	3	4	4	4	5	5	3	3	5	5	1	4	4	4	5	3	4	4	4	4	2	4	4	4	3	3	3	2	4	2	4	4	4	4	4	4
41	Male	IT Specialist	7	2	4	4	5	5	5	4	5	2	4	4	4	2	4	4	5	5	2	2	5	4	5	5	3	5	4	4	4	4	2	5	5	4	4	4	3	2	2	2	4	3	3	4
42	Male	Software developer	5	5	20	5	5	5	4	4	4	4	4	5	5	5	5	5	4	4	1	2	4	4	5	2	2	4	4	2	2	2	4	4	5	2	2	4	1	4	1	1	1	1	1	1
43	Male	Project manager	8	3	4	3	5	5	4	3	5	2	4	4	4	5	5	5	5	2	4	4	1	5	5	5	2	4	5	5	5	4	5	3	5	5	3	1	4	2	4	4	5	2	3	5
44	Female	Software specialist	0.5	1	4	4	5	5	4	2	3	3	4	3	5	5	5	2	1	5	3	1	5	3	4	5	5	1	4	3	5	5	5	2	4	2	4	5	5	4	1	5	5	5	5
45	Male	Software developer	12	12	4	4	4	4	3	4	4	4	3	5	5	4	3	3	3	2	4	3	3	4	3	4	4	3	4	3	3	3	3	3	4	3	3	3	3	4	2	3	4	4	4
46	Male	Software developer	11	6	6	10	4	5	5	4	4	3	2	4	3	5	4	2	5	4	3	1	4	5	5	4	4	5	4	4	4	4	2	4	4	4	5	3	3	3	4	3	4	4	4
47	Male	Software developer	4	1	6	2	5	5	5	4	5	5	2	5	4	2	5	4	5	5	2	1	5	4	5	2	2	1	4	3	2	2	5	2	4	5	5	2	3	4	4	5	2	4	4
48	Male	Software Tester	0	0	11	3	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
49	Female	Software developer	3	1	4	3	5	5	4	3	4	5	3	4	4	2	5	2	2	4	3	2	4	2	5	5	1	5	4	4	4	5	4	4	2	2	4	2	2	2	2	2	1	3	3
50	Male	Software specialist	4	1	12	10	5	5	4	4	5	5	4	4	5	3	4	5	4	5	3	2	4	3	4	2	2	4	4	4	2	3	5	4	4	5	4	4	4	4	4	5	5	5	4
51	Male	Software specialist	3	2	11	11	5	5	4	5	5	5	5	3	4	3	3	5	4	2	3	3	5	2	5	2	5	5	4	5	1	1	5	1	5	3	3	3	5	1	1	4	1	1	1
52	Male	Software developer	3	3	4	3	5	5	4	4	5	4	2	5	5	4	5	3	4	5	2	1	4	4	5	5	5	5	5	5	5	5	2	5	5	5	4	4	4	5	4	2	3	4	4	4
53	Male	Software Tester	1	1	7	4	5	4	5	4	5	5	3	4	4	2	4	1	2	5	2	2	5	5	5	5	4	5	5	5	5	4	2	5	5	5	5	5	5	4	4	4	5	5	3	4
54	Male	Software specialist	1	1	3	3	5	5	4	4	4	3	4	3	5	2	4	4	4	4	2	2	4	3	4	5	4	4	4	5	5	3	5	3	5	4	4	4	4	3	2	2	3	4	4
55	Male	IT Specialist	10	1	4	4	5	5	5	4	3	5	2	4	4	5	5	5	1	3	4	2	1	5	4	5	5	5	5	5	4	2	4	4	5	5	4	4	1	2	1	5	2	2	5
56	Female	Software developer	0	0	5	5	5	5	4	4	5	4	3	4	5	2	4	1	5	4	3	1	4	4	5	5	4	5	5	5	5	5	2	4	5	5	5	4	4	4	2	2	2	4	5	5
57	Male	IT Specialist	8	1	3	3	5	5	4	3	3	4	3	4	4	4	4	3	4	4	3	3	5	5	5	5	4	4	4	5	5	5	4	5	4	3	4	3	3	5	3	3	5	3	5
58	Female	IT Specialist	3	0	5	5	4	4	4	3	4	4	4	3	4	3	2	2	4	5	4	2	3	3	4	4	3	4	3	4	4	3	4	4	4	3	4	4	4	3	3	3	2	3	3	4
59	Male	Software specialist	2	1	15	13	5	5	5	4	5	4	3	4	4	4	5	3	1	3	1	1	5	5	5	3	4	5	4	5	5	4	4	4	5	2	2	2	2	4	2	4	3	2	4
60	Male	Software specialist	3	2	15	10	5	4	4	3	4	3	2	4	2	4	2	5	3	4	2	2	4	4	4	5	4	5	4	3	3	5	4	2	4	2	1	3	4	2	1	3	4	5	3
61	Female	Software developer	6	6	2	2	5	5	5	5	4	5	3	5	3	4	5	4	5	5	4	2	5	3	5	4	2	4	5	4	4	1	5	5	5	4	4	4	4	3	1	2	1	1	4
62	Male	Software developer	5	3	12	5	5	5	4	4	4	5	2	4	5	3	5	4	4	5	1	1	5	4	5	5	4	4	5	4	5	4	4	4	4	3	3	4	1	4	4	3	3	4	4
63	Male	Project manager	25	5	4	4	5	5	5	4	5	4	2	4	4	5	4	4	5	5	5	3	5	5	5	5	4	5	5	5	5	4	5	4	5	4	5	1	4	1	5	2	2	3	1	1	5
64	Male	Project manager	15	2	11	4	5	5	4	4	4	4	4	4	5	5	3	4	5	4	4	4	2	4	4	4	4	4	4	5	5	5	4	3	5	4	5	3	4	3	3	4	4	3	4	4	3
65	Male	Software Tester	4	3	5	5	5	2	5	3	5	4	2	4	4	5	5	4	3	4	4	2	4	4	4	4	5	4	4	4	3	3	4	4	3	4	3	3	3	3	2	4	4	4	4
66	Male	Software developer	9	1	3	3	5	5	4	4	4	5	4	4	3	3	4	3	5	5	3	2	4	2	5	4	5	5	5	4	4	4	4	5	5	5	5	5	5	5	4	5	4	4	4
67	Male	System Analyst	15	5	1	1	5	4	4	4	5	3	5	4	4	5	3	3	5	4	1	5	5	5	5	3	5	4	4	3	5	5	1	5	3	4	4	4	3	3	4	1	3	3	4	5
68	Male	Software developer	1	1	5	3	3	4	3	4	5	4	3	3	4	4	5	4	2	5	1	2	5	4	4	4	3	5	4	4	4	3	5	3	4	3	4	3	4	3	3	3	4	4	4
69	Male	Project manager	9	2	8	3	5	5	2	1	2	1	2	4	4	5	4	5	1	5	5	2	5	2	5	5	4	4	5	5	4	2	3	4	5	4	5	4	2	5	1	5	2	2	4

ID NO	SEX	JOB TITLE	WORK _YEAR	WORK _THIS _FIRM	TEAM_ SIZE	IDEAL_ TEAM_ SIZE	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	X14	X15	X16	X17	X20	X21	X22	X23	X24	X25	X26	X27	X28	X29	X30	X31	X32	X33	X34	X35	X36	X37	X38	X39	X40	X41
70	Male	Software developer	3	1	4	4	5	4	4	4	4	5	3	4	4	5	5	4	5	5	3	2	5	4	5	5	3	4	3	4	4	2	4	4	5	4	4	4	5	3	2	2	4	4	4
71	Male	Software specialist	1	0	5	4	5	5	4	5	5	5	4	4	4	5	3	5	5	5	4	1	5	4	5	5	4	5	5	5	4	4	4	4	5	4	4	4	3	4	1	4	3	5	5
72	Female	System Analyst	22	22	44	13	5	5	5	1	1	4	2	3	3	5	5	5	3	3	2	5	5	5	5	3	2	3	4	4	5	5	5	4	4	5	1	1	1	1	2	4	3	4	3
73	Male	IT Specialist	18	2	5	4	5	5	4	4	4	4	3	4	4	4	3	4	3	4	4	3	4	4	4	4	3	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
74	Male	Software developer	3	1	2	1	5	5	4	4	4	2	2	4	5	4	4	2	5	5	1	3	4	4	4	5	4	4	4	5	4	4	5	4	4	5	5	3	4	4	4	2	2	4	4
75	Male	Software specialist	0	0	21	21	5	5	4	5	5	2	2	4	5	4	4	5	4	5	5	2	4	4	5	5	4	5	4	5	4	4	4	3	4	4	4	4	3	4	4	3	4	4	4
76	Female	Team Leader	11	3	5	2	5	4	4	2	4	2	2	2	4	3	4	2	4	4	2	2	4	4	5	4	4	5	4	2	2	5	2	5	4	2	4	2	4	3	2	2	4	4	2
77	Female	Software Tester	1	1	7	4	5	5	4	3	5	5	4	4	4	2	4	4	4	5	4	1	5	3	5	5	4	5	4	4	5	5	4	5	5	3	4	3	2	2	4	5	5	4	4
78	Male	Project manager	21	13	2	2	4	5	3	2	5	4	2	2	4	4	5	5	4	4	2	4	4	4	4	1	3	5	4	4	1	1	4	4	4	4	4	4	4	4	2	2	2	4	
79	Male	IT Specialist	8	1	10	3	4	3	4	4	4	4	4	4	4	4	2	4	4	4	2	4	4	4	4	3	2	4	4	4	4	4	4	4	4	4	4	4	4	4	2	2	4	4	4
80	Male	Software specialist	3	2	7	5	5	5	4	4	5	4	4	4	4	3	4	4	5	4	4	3	5	4	4	3	4	4	4	4	2	4	4	2	3	5	4	4	4	3	2	5	3	3	4
81	Male	Software developer	3	1	6	3	4	5	5	5	5	5	3	4	5	4	3	5	4	5	3	1	4	4	5	4	5	5	4	4	2	4	5	5	4	4	5	5	3	3	2	2	2	3	3
82	Male	Project manager	10	2	5	5	4	4	5	4	5	2	4	4	5	4	4	5	4	4	2	2	4	2	4	5	4	4	4	3	2	2	4	4	4	2	2	2	3	3	1	3	2	2	3
83	Male	Software developer	13	1	8	7	5	4	4	4	4	4	4	4	4	4	4	2	4	4	4	2	4	4	4	4	4	4	4	4	3	2	4	4	4	4	4	4	4	2	2	4	4	4	4
84	Male	Software Tester	6	2	3	3	4	4	3	3	4	4	3	4	4	5	5	4	5	2	2	3	4	5	4	4	3	3	4	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
85	Male	Team Leader	12	1	10	7	5	5	4	4	5	4	4	4	5	4	4	4	3	4	4	3	3	4	4	4	4	4	5	5	5	5	5	4	5	4	4	4	4	4	3	4	2	3	4
86	Male	Software Tester	5	1	5	4	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
87	Female	Software Tester	3	2	5	5	5	5	5	5	5	5	5	4	5	3	5	5	3	4	3	3	4	5	5	5	5	5	5	5	5	5	2	3	4	4	4	4	4	5	4	2	4	4	4
88	Male	IT Specialist	6	1	8	1	5	4	3	4	4	5	4	4	4	5	5	4	2	4	4	4	5	3	4	2	2	2	4	4	4	2	2	4	4	5	4	2	4	3	2	2	2	4	4
89	Male	Project manager	10	1	5	3	5	5	3	3	4	4	2	3	4	3	4	3	4	4	3	2	4	3	4	5	4	4	4	4	3	3	3	4	3	3	4	3	3	3	5	2	3	4	
90	Female	Software developer	0.5	1	5	5	4	4	4	4	4	4	3	3	4	4	3	5	3	5	4	3	4	4	4	4	5	3	4	4	4	2	4	4	5	4	4	3	4	3	4	2	4	3	4
91	Female	IT Specialist	1	1	3	3	5	5	4	4	4	4	4	4	4	4	4	5	5	4	4	5	5	3	4	4	4	4	5	3	4	4	4	4	4	4	4	4	4	2	3	2	4	4	
92	Male	Software architecture	30	10	19	12	5	5	5	3	5	5	4	4	5	5	4	4	5	4	1	5	5	5	5	3	3	5	4	5	1	4	4	5	5	1	1	1	1	3	1	4	3	3	1
93	Male	Software developer	11	2	7	2	5	5	4	5	4	3	4	5	4	5	4	4	4	5	4	2	5	4	5	4	4	5	4	4	5	2	5	2	4	3	5	4	2	3	1	3	4	4	3
94	Male	Software Tester	4	1	5	5	5	4	3	5	5	5	2	3	5	4	5	4	4	5	3	3	5	5	4	4	4	5	4	4	2	4	5	4	4	4	4	3	4	4	2	4	4	4	4
95	Male	IT Specialist	8	2	5	3	4	4	4	3	4	3	3	4	3	4	4	2	4	4	3	3	3	2	2	4	3	3	2	3	4	4	3	4	2	4	4	4	4	5	4	2	3	2	3
96	Male	Software specialist	1	1	7	7	4	5	4	4	4	4	3	4	2	4	5	5	5	5	2	3	4	5	5	5	4	4	4	3	4	4	5	5	5	4	5	4	5	2	2	2	2	3	1
97	Male	IT Specialist	12	2	5	5	5	5	4	4	4	4	3	4	5	2	4	5	2	5	5	2	3	4	5	5	5	5	5	4	5	3	4	5	4	4	2	4	4	3	3	3	3	3	3
98	Male	Software specialist	3	3	4	4	5	4	4	4	4	5	2	3	5	5	4	4	4	5	5	2	1	5	4	4	5	5	4	4	5	5	2	4	5	4	4	4	4	5	4	2	3	4	4
99	Female	Software specialist	1	1	5	2	5	5	4	4	4	4	3	3	4	3	4	4	3	5	5	2	4	4	4	4	4	4	5	4	5	5	4	4	4	4	4	5	3	3	3	4	3	3	4
100	Male	IT Specialist	8	2	4	4	5	5	4	5	5	5	5	5	4	5	5	5	5	5	4	1	5	4	5	5	4	5	5	5	5	5	5	4	5	4	5	4	4	5	2	4	2	5	5
101	Male	Software specialist	1	1	3	3	5	4	4	4	3	4	4	4	3	2	4	3	3	5	4	3	3	4	5	4	4	2	4	4	4	4	5	4	3	4	3	3	3	2	3	4	3	3	3
102	Female	Software Tester	2	2	6	6	4	5	3	4	5	4	4	4	5	3	2	4	3	5	5	2	5	4	5	5	4	4	4	4	5	4	5	5	5	4	4	5	4	3	3	4	3	4	4
103	Male	Project manager	9	1	6	4	5	5	4	4	5	4	3	4	5	5	5	3	4	4	3	2	4	4	4	4	3	4	4	4	4	3	4	4	4	4	2	2	4	2	2	2	2	4	4
104	Male	Software developer	1	1	3	3	5	5	5	4	5	5	4	4	4	3	4	4	4	4	3	4	4	4	4	4	4	3	4	4	5	4	4	4	4	5	4	4	2	4	2	2	2	4	4
105	Male	Software developer	4	2	5	3	5	4	5	3	4	4	3	3	5	3	5	3	4	5	2	2	5	4	5	5	5	5	4	5	5	5	5	5	5	5	4	4	4	5	4	2	4	2	4
106	Female	Software Tester	2	1	13	7	4	5	4	4	4	4	3	3	4	2	3	2	3	5	4	2	2	4	4	5	4	4	4	4	4	3	5	4	4	3	3	4	3	3	4	2	4	4	4
107	Male	Software developer	4	3	13	5	5	5	4	2	4	4	3	3	5	4	4	5	3	5	4	2	1	4	2	4	5	4	5	4	4	3	4	5	5	5	5	2	5	4	3	4	2	4	4
108	Male	IT Specialist	5	1	5	2	5	5	4	5	5	4	3	2	4	4	2	5	5	5	4	5	5	4	5	4	2	3	4	3	3	4	4	2	4	5	5	5	3	4	1	5	5	4	4
109	Male	Software developer	5	1	15	10	5	5	4	3	3	2	2	3	5	5	5	2	2	5	4	2	5	5	5	4	3	5	3	1	1	5	5	4	4	4	2	2	2	3	1	2	2	4	4
110	Male	Software Tester	7	1	6	6	5	4	4	4	4	5	4	4	5	4	5	3	5	5	4	2	5	4	5	5	4	4	5	4	4	5	4	5	5	5	4	4	4	4	4	2	4	4	4
111	Female	Software developer	4	0	20	20	4	4	3	4	4	4	3	3	4	4	4	4	4	4	4	3	3	3	4	4	4	4	4	4	3	2	4	4	4	4	4	4	3	4	2	2	4	4	3
112	Female	Software Tester	7	2	4	4	5	4	4	4	5	3	3	4	4	3	4	3	4	4	4	3	4	2	4	4	3	5	3	4	4	4	4	3	4	3	4	2	3	2	1	3	2	2	3
113	Female	Software specialist	2	2	4	4	5	5	4	4	4	4	4	4	4	4	4	4	4	5	4	1	4	4	4	4	4	4	4	4	4	4	4	4	3	4	3	4	4	4	4	1	4	4	4
114	Female	System Analyst	7	5	8	8	5	4	4	4	5	5	4	4	4	3	4	5	3	4	3	2	4	4	4	4	3	4	4	4	3	4	3	4	2	2	2	3	2	3	3	5	3	3	3
115	Male	Software developer	4	1	10	3	5	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
116	Female	System Engineer	3	2	7	3	4	4	3	4	4	4	3	4	4	3	4	4	4	4	4	4	4	4	4	4	4	5	2	3	4	4	2	4	5	4	5	5	5	3	4	4	4	5	5
117	Male	Team Leader	20	3	5	4	4	4	4	4	4	3	3	4	4	4	4	3	3	4	3	4	3	3	4	4	3	4	4	3	3	4	4	4	4	3	4	3	3	2	2	4	3	4	4
118	Female	Software specialist	2	2	40	40	5	5	4	4	5	5	3	4	5	5	5	5	5	5	3	2	5	5	5	4	4	5	5	4	2	3	4	4	4	4	4	4	5	4	2	4	5	4	5
119	Male	IT Specialist	10	1	6	6	5	4	4	4	5	5	3	3	5	4	5	5	4	5	3	1	5	2	4	5	5	5	5	5	2	1	4	5	5	2	4	3	5	3	5	2	4	4	3
120	Male	Team Leader	7	1	2	2	5	5	4	4	5	4	2	4	4	5	5	4	4	4	2	2	4	2	4	4	4	4	4	4	4	2	5	4	4	4	2	4	2	2	2	2	3	4	3
121	Female	Software Tester	1.5	2	13	13	4	5	4	2	4	4	4	4	4	5	4	3	2	5	1	4	4	5	5	5	4	5	5	5	5	5	2	4	2	4	2	4	4	2	4	4	2	4	4
122	Male	Software developer	10	8	9	5	5	5	4	4	4	4	4	4	4	4	4	3	4	5	4	3	4	3	4	5	5	4	4	4	4	3	4	4	4	4	5	4	4	4	4	4	3	3	4
123	Male	System Engineer	26	3	12	5	5	5	4	4	4	2	3	4	5	4	5	5	2	4	2	1	4	4	4	4	4	4	4	4	4	2	2	2	4	4	4	2	2	4	1	1	4	4	2
124	Male	Software developer	8	5	8	5	5	3	4	4	4	4	4	5	4	5	2	3	4	4	3	5	4	5	5	4	4	4	5	4	4	3	5	3	4	4	4	4	4	2	4	4	4	5	
125	Male	System Analyst	7	3	5	2	5	5	4	4	4	5	3	4	5	5	5	4	5	5	4	2	4	4	4	5	5	4	4	4	5	5	3	4	4	2	2	2	4	3	3	4	4	4	4
126	Male	Team Leader	13	2	30	5	5	5	4	3	4	5	3	4	4	3	5	5	4	4	4	2	4	3	5	5	4	4	4	5	5	3	5	4	5	4	5	4	5	4	3	4	3	3	3
127	Male	Software developer	4	4	6	6	5	5	5	5	5	1	4	4	5	5	5	4	5	5	1	1	4	5	5	5	5	5	5	5	5	5	4	5	4	5	2	2	2	4	1	2	2	4	5
128	Female	Team Leader	10	5	2	2	4	5	3	3	3	4	2	3	4	4	3	4	3	4	2	3	4	4	4	4	4	4	4	4	4	3	4	4	4	4	3	2	2	3	4	1	3	3	4
129	Male	Project manager	11	1	5	3	5	4	4	4	5	4	4	2	4	2	4	3	2	5	3	2	4	4	5	4	4	4	4	4	4	4	4	4	3	2	1	2	2	2	3				
130	Female	Software developer	8	2	27	10	5	5	5	4	5	4	5	5	5	5	5	4	5	5	2	2	5	2	5	5	4	4	5	5	2	2	5	4	5	4	2	2	5	2	1	2	3	4	5
131	Male	Software developer	4	1	5	2	4	5	3	3	3	4	4	4	3	4	4	4	4	4	3	3	4	4	4	4	3	3	3	4	4	3	3	3	3	4	2	2	2	2	2	3	2	3	3
132	Male	Software specialist	5	5	6	6	5	4	4	4	5	5	4	4	4	5	5	4	4	5	3	3	5	3	5	5	3	2	4	5	4	3	4	4	5	4	4	4	2	4	1	2	3	5	4
133	Male	Software developer	5	2	9	5	4	4	4	4	4	5	4	4	4	4	4	2	3	5	2	2	4	3	4	4	4	4	4	4	4	4	2	4	4	4	3	4	3	4	3	2	4	4	4
134	Male	Project manager	7	1	3	3	5	5	5	3	4	5	4	4	4	3	4	2	1	5	1	1	4	3	4	5	4	2	4	4	4	5	5	5	4	5	4	2	1	4	4	5	5		
135	Male	System Engineer	25	2	10	10	5	2	2	2	4	3	2	4	4	3	5	4	3	5	3	2	5	4	4	4	3	5	3	4	4	3	3	4	5	4	3	4	5	3	2	3	4	4	4
136	Male	Software specialist	1	1	5	5	4	5	3	4	5	5	3	4	4	5	5	4	3	5	4	2	2	2	4	4	3	3	5	4	4	4	5	3	4	4	4	3	3	3	4	4	4	4	3
137	Male	Software developer	13	2	8	4	5	5	3	3	5	4	3	4	4	4	5	3	4	4	4	2	4	3	4	4	3	5	4	4	4	5	2	4	4	5	3	4	3	4	4	1	1	3	4
138	Male	Software specialist	3	1	8	8	5	5	5	4	5	5	4	4	4	4	4	4	4	4	3	4	4	2	4	4	5	4	4	4	4	1	5	4	4	4	3	4	4	4	2	4	3	4	4

ID NO	SEX	JOB TITLE	WORK _YEAR	WORK _THIS _FIRM	TEAM_ SIZE	IDEAL_ TEAM_ SIZE	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	X14	X15	X16	X17	X20	X21	X22	X23	X24	X25	X26	X27	X28	X29	X30	X31	X32	X33	X34	X35	X36	X37	X38	X39	X40	X41
139	Female	IT Specialist	6.5	1	5	5	5	4	5	3	5	5	5	5	5	4	5	4	5	5	3	4	4	5	5	5	4	5	5	5	2	4	5	5	5	4	3	3	3	3	1	3	4	4	5
140	Male	Software specialist	2	2	6	5	4	3	5	4	4	4	3	4	4	5	5	4	3	4	3	4	4	3	4	4	3	5	4	3	4	3	4	5	5	4	5	5	4	3	4	3	5	4	4
141	Male	Project manager	10	1	3	2	5	5	5	2	5	5	4	4	4	3	4	4	5	4	3	5	5	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
142	Female	Team Leader	16	2	13	6	5	5	4	4	4	4	4	4	4	4	5	4	5	4	2	1	4	2	4	4	3	4	4	5	4	2	4	4	4	4	4	1	4	2	2	2	2	4	3
143	Male	IT Specialist	11	2	8	8	4	4	4	2	4	4	2	4	4	3	4	3	2	5	2	1	4	4	4	4	4	4	4	4	4	4	4	4	4	2	2	1	2	2	1	4	3	4	4
144	Female	System Analyst	8	1	14	8	4	4	4	3	4	4	2	2	4	4	4	2	2	4	4	2	4	4	4	4	4	5	4	4	2	2	4	3	4	2	4	3	3	3	2	2	4	4	4
145	Male	IT Specialist	9	2	6	4	5	4	2	4	4	4	2	4	4	4	4	4	4	4	4	2	4	4	4	5	5	4	4	4	4	2	4	5	4	4	4	4	4	4	4	4	4	4	4
146	Female	Software Tester	1	1	8	5	4	4	3	4	4	5	4	4	5	3	4	4	5	5	3	3	4	4	5	5	3	5	4	4	3	4	4	5	4	4	3	3	4	3	3	3	3	4	4
147	Female	Software Tester	3	8	6	6	4	4	4	3	4	5	4	4	4	5	5	4	3	4	4	2	2	4	4	3	3	4	4	4	4	3	4	4	4	4	4	4	3	2	1	3	3	4	3
148	Male	Software developer	10	12	5	5	4	5	2	4	4	5	5	2	4	4	5	4	5	5	2	2	4	2	5	5	4	4	3	4	4	2	4	4	5	4	4	4	3	2	2	2	4	5	2
149	Male	Software developer	5	3	7	7	5	4	3	4	4	5	2	4	5	3	2	4	5	5	2	1	4	2	5	5	4	4	4	5	2	1	3	2	4	4	4	2	4	2	1	2	2	4	2
150	Female	System Analyst	3	1	10	5	5	5	5	4	4	4	4	4	5	4	4	4	4	4	4	3	4	4	4	4	3	2	4	4	4	5	4	4	4	3	3	3	3	4	2	4	4	4	4
151	Female	IT Specialist	0	1	4	2	4	2	4	5	5	4	4	4	4	3	4	2	3	4	2	2	4	3	4	4	4	4	4	4	3	4	3	4	4	2	2	2	2	2	2	2	4	4	4
152	Male	IT Specialist	1	1	5	5	4	4	4	4	4	2	2	2	5	4	4	2	2	4	2	4	4	4	4	4	3	4	4	4	5	4	3	4	2	4	4	4	2	4	2	2	3	4	2
153	Male	IT Specialist	9	2	6	5	5	2	4	3	3	5	2	4	4	1	3	3	4	4	3	5	5	4	4	4	2	4	3	4	4	2	4	2	4	3	1	1	3	2	1	1	2	3	4
154	Male	System Analyst	2	1	10	10	4	5	2	4	5	4	4	4	4	4	5	4	3	5	3	1	5	3	2	5	4	5	5	5	5	1	2	4	5	5	2	4	3	2	2	5	4	5	5
155	Male	Software developer	8	4	3	3	5	4	3	2	4	5	4	4	5	3	4	4	4	2	1	4	5	2	2	4	5	4	3	4	5	2	4	1	2	3	2	3	3	3	2	4	4	4	4
156	Male	System Engineer	10	2	7	3	4	4	5	4	3	2	3	4	3	2	3	2	3	4	3	3	3	4	4	3	4	4	4	4	3	4	4	3	4	4	4	3	3	3	4	4	3	4	3
157	Male	Team Leader	1	1	3	3	5	5	4	5	5	5	4	4	5	3	5	2	4	5	4	5	4	3	5	5	4	5	4	5	5	3	5	5	5	5	5	4	4	2	2	5	2	5	5
158	Male	Software developer	4	3	7	3	5	5	3	3	4	4	3	3	5	4	4	4	2	4	2	4	5	4	4	4	5	5	5	4	4	5	4	4	5	4	4	3	3	4	1	4	3	4	4
159	Male	Software developer	1.2	1	6	2	5	5	2	3	4	2	5	5	5	1	4	5	5	5	1	1	5	5	5	5	1	5	3	5	5	5	3	5	5	4	4	4	4	4	1	1	1	1	3
160	Male	Software Tester	8	4	6	3	5	5	5	4	4	4	4	5	4	4	5	2	5	5	5	1	5	2	4	5	5	5	5	4	5	4	5	4	5	4	4	4	2	1	5	2	4	5	5
161	Male	Software developer	7	7	6	2	5	5	4	4	4	3	2	4	4	5	4	4	4	4	3	2	4	4	4	4	4	4	4	4	4	2	4	4	4	5	4	4	4	2	1	4	4	4	4
162	Male	Software developer	3.5	1	6	2	5	5	4	5	5	5	4	4	4	4	5	4	4	5	4	2	4	2	4	3	4	5	4	4	3	5	3	4	5	5	3	4	4	5	1	5	4	4	3
163	Female	IT Specialist	0	1	5	3	5	5	5	5	5	4	3	5	4	5	5	3	5	2	2	5	5	5	5	4	5	5	5	5	5	5	4	5	5	5	5	5	2	4	2	5	5	5	5
164	Female	Software specialist	2	2	2	2	5	4	3	4	5	4	3	2	5	5	3	5	4	2	2	2	4	4	5	4	3	4	5	4	4	5	4	4	4	3	4	2	3	4	4	2	2	4	4
165	Male	Software specialist	1	1	4	4	5	4	4	4	4	3	3	4	4	3	4	5	4	4	4	2	4	4	5	5	3	2	4	4	4	3	2	4	4	4	3	2	1	4	2	2	2	4	4
166	Male	Software specialist	3	2	8	3	5	5	4	4	5	4	4	4	5	4	4	3	4	4	3	2	4	4	4	4	4	4	4	4	5	4	4	4	4	3	4	3	3	4	2	4	3	4	4
167	Male	Project manager	6	1	7	7	5	5	4	4	4	5	4	4	4	4	5	4	3	4	2	1	5	4	5	4	4	4	4	4	2	4	4	5	5	5	4	4	4	4	2	2	3	4	4
168	Female	System Analyst	7	1	40	30	4	4	4	3	4	4	2	3	4	4	4	3	3	3	3	2	4	3	4	4	3	4	4	4	3	4	4	4	4	4	4	3	4	2	4	3	3	4	4
169	Male	Software developer	2	2	6	6	5	5	4	4	4	4	4	4	4	4	4	2	4	4	3	3	4	4	4	4	4	4	2	4	4	4	4	5	4	4	4	4	4	4	4	2	3	2	3	4
170	Female	Software specialist	1	1	12	5	5	5	5	5	5	4	4	4	5	4	4	4	4	4	4	4	4	4	5	5	4	4	4	4	3	4	4	4	4	5	4	4	4	4	1	3	2	4	4
171	Male	System Analyst	7	3	8	6	2	2	2	1	2	4	2	2	1	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
172	Female	Team Leader	11	10	20	20	5	5	4	4	4	5	4	4	3	3	5	2	2	5	2	3	5	3	5	5	3	5	4	5	4	3	4	5	5	5	5	5	5	3	2	5	3	3	5
173	Male	Software developer	3	3	13	13	4	4	4	2	4	4	3	4	3	4	4	3	4	4	2	2	2	3	4	4	3	4	3	4	4	3	4	4	4	4	4	4	4	4	4	4	4	4	4
174	Female	Software developer	5	1	8	4	5	5	5	3	5	5	4	4	4	4	4	3	5	4	3	2	5	5	4	5	4	5	5	5	5	4	4	4	5	4	3	4	2	4	2	2	4	4	5
175	Female	Software Tester	3	1	14	5	4	4	3	5	5	3	3	4	4	2	4	2	1	5	3	1	2	4	4	5	3	5	4	5	5	5	5	5	5	4	4	4	5	5	2	3	3	4	4
176	Male	Software developer	14	4	6	3	5	5	4	4	4	3	3	3	4	3	4	3	4	4	3	3	4	4	4	4	4	4	4	4	4	4	4	4	3	4	4	4	4	4	4	4	4	4	4
177	Male	Software specialist	3	1	9	9	5	4	3	4	4	5	4	4	4	3	5	4	2	5	4	2	4	3	4	5	4	4	4	4	5	4	5	4	4	3	3	2	4	4	2	4	3	4	4
178	Male	System Analyst	4	2	8	8	5	5	5	5	5	3	3	1	5	5	5	5	5	5	5	1	5	5	5	5	5	5	5	5	5	3	5	5	5	5	5	5	5	1	5	1	5	5	5
179	Male	IT Specialist	9	2	7	4	5	5	3	3	4	3	2	4	5	3	2	4	4	5	1	2	4	4	5	4	4	5	3	4	3	2	4	4	4	3	4	4	3	3	2	3	3	3	4
180	Female	Software Tester	1	1	8	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	3	4	4	5	5	5	4	5	3	5	5	4	5	4	4	4	4	4	5
181	Female	Software Tester	4	1	8	3	5	5	5	4	5	5	4	4	4	4	4	3	4	4	1	1	2	4	5	5	3	5	5	5	4	5	3	5	5	4	5	4	4	4	2	3	3	4	4
182	Female	Software Tester	4	1	5	3	5	5	5	4	5	4	4	4	5	5	5	4	4	3	3	2	4	5	5	4	1	5	5	5	5	5	3	4	4	4	4	3	3	3	4	3	2	4	4
183	Male	Project manager	13	2	15	5	5	5	4	3	4	5	5	5	4	3	4	4	4	5	5	2	4	4	5	5	4	5	5	5	5	4	5	4	5	4	5	4	5	3	3	4	3	4	5
184	Male	Software developer	10	3	3	1	5	5	4	4	4	3	4	4	4	5	5	2	5	5	5	1	1	2	5	5	5	5	5	5	5	5	5	5	5	5	3	4	2	2	4	1	1	1	5	5
185	Male	Software Tester	2	5	6	6	5	5	5	4	4	4	5	4	3	4	4	5	3	5	3	2	4	4	4	4	5	5	4	4	4	4	5	5	4	4	4	4	4	4	2	5	3	3	4
186	Male	IT Specialist	2	4	5	3	5	4	3	3	4	3	2	4	4	3	4	4	3	4	4	4	4	4	5	4	2	4	4	4	4	4	5	4	4	4	4	4	4	4	3	5	3	4	4	4
187	Female	Software developer	11	9	10	4	4	4	4	4	4	5	3	3	5	4	4	4	4	5	5	3	5	4	5	5	3	5	4	5	2	5	5	4	4	3	4	4	4	4	1	3	4	4	4
188	Male	IT Specialist	7	1	4	4	4	4	4	2	4	4	4	4	4	5	4	2	4	4	4	2	4	4	4	4	4	4	4	4	3	4	4	3	4	4	4	3	2	4	2	4	4	4	4
189	Male	Project manager	12	1	6	4	5	5	4	3	4	4	3	4	3	4	3	4	2	3	2	2	4	4	5	5	3	4	4	4	4	4	4	5	4	4	4	4	3	4	2	1	4	4	4
190	Female	System Engineer	3	2	6	6	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	3	4	4	5	5	5	5	4	4	3	4	4	5	4	5	5
191	Male	Software developer	4	4	6	2	5	5	4	3	4	3	4	4	5	4	4	4	4	4	4	3	5	4	5	5	4	3	4	4	4	2	4	3	4	4	3	4	4	3	3	4	3	4	4
192	Male	Team Leader	4	2	3	1	4	4	4	4	5	4	2	4	5	5	5	5	4	5	2	1	4	4	5	5	3	5	5	5	5	4	3	5	2	4	4	2	2	2	2	5	3	3	4

Appendix D

Cronbach Alpha Calculations

$$\alpha_{(39questions)} = \frac{39}{38}\left(\frac{162.02 - 30.47}{162.02}\right) = 0.83 \tag{D.1}$$

$$\alpha_{Productivity(17questions)} = \frac{17}{16}\left(\frac{34.25 - 12.50}{34.25}\right) = 0.68 \tag{D.2}$$

$$\alpha_{SocialProductivity(12questions)} = \frac{12}{11}\left(\frac{25.35 - 8.50}{25.35}\right) = 0.73 \tag{D.3}$$

$$\alpha_{SocialCapital(10questions)} = \frac{10}{9}\left(\frac{30.23 - 9.46}{30.23}\right) = 0.76 \tag{D.4}$$

Appendix E

Pilot Study Card Game Game Data

INTERVIEW RESPONSES:

ID-NO	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15	Q16	Q17	Q18	Q19	Q20	Q21	Q22	Q23	Q24	Q25	Q26	Q27	Q28	Q29	Q30	Q31	Q32	Q33	Q34	Q35	Q36
1	1	1	2	1	2	1	2	1	2	1	2	2	1	1	2	2	2	1	2	1	1	1	1	2	2	2	1	1	2	2	1	2	2	2	2	1
2	1	1	1	2	2	2	1	1	2	1	1	2	2	1	1	2	2	1	2	2	1	1	1	2	2	1	1	1	2	1	1	2	1	2	2	2
3	2	1	2	2	2	2	2	2	2	1	1	2	1	1	1	2	1	2	2	2	2	1	1	2	1	2	1	1	2	1	1	2	2	2	1	2
4	1	1	1	2	2	2	2	1	2	1	2	2	2	2	1	2	1	2	2	2	1	1	2	2	2	2	2	1	2	1	2	2	1	1	2	2
5	1	1	1	2	2	2	2	1	2	1	2	1	2	1	1	2	1	1	2	2	1	1	2	2	2	2	2	2	2	1	1	2	2	1	1	1
6	2	2	1	2	1	1	2	2	1	1	2	1	1	1	2	2	2	1	2	2	2	2	2	2	2	2	1	1	2	1	1	2	1	1	2	2
7	1	1	1	2	2	1	2	1	2	1	2	2	2	2	1	2	1	2	2	2	2	1	1	2	2	2	2	2	2	1	1	2	2	1	2	2
8	1	2	2	1	2	1	1	1	2	2	1	2	1	1	1	1	2	1	2	1	1	1	1	2	2	1	1	1	2	1	1	2	1	2	1	1
9	2	2	2	1	1	2	1	1	2	2	1	1	1	2	1	2	2	1	1	1	1	1	2	2	1	2	1	1	1	2	2	2	1	2	1	2
10	2	1	2	1	2	1	2	1	2	2	1	1	1	2	1	2	1	1	2	2	2	2	1	1	2	1	1	1	2	1	2	1	1	2	1	1
11	2	1	1	2	2	2	1	2	1	2	1	2	1	2	1	2	2	2	2	2	2	2	1	2	2	1	1	2	2	2	1	2	1	2	2	2
12	1	2	1	2	1	2	2	1	2	2	1	1	1	2	2	2	2	1	2	1	2	1	2	2	1	2	1	2	2	1	1	2	1	2	2	1
13	1	1	2	2	2	2	1	1	2	2	1	1	1	1	1	2	1	1	2	1	1	1	2	2	1	2	1	1	2	2	1	1	2	2	2	2
14	2	1	2	1	1	2	1	1	2	2	1	1	1	1	2	1	1	1	1	2	2	2	1	2	1	2	1	1	1	2	1	1	1	1	2	2
15	1	1	1	2	2	2	2	1	1	1	1	2	1	1	1	1	1	1	2	1	1	2	1	1	2	2	1	1	1	1	2	2	2	2	1	1

RE-INTERVIEW RESPONSES:

ID-NO	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15	Q16	Q17	Q18	Q19	Q20	Q21	Q22	Q23	Q24	Q25	Q26	Q27	Q28	Q29	Q30	Q31	Q32	Q33	Q34	Q35	Q36
1	2	2	1	1	2	1	1	1	2	1	1	2	1	2	2	2	2	1	2	1	1	2	2	2	2	2	1	1	2	2	2	1	2	1	1	2
2	2	1	2	2	2	2	1	1	1	1	1	2	2	1	1	2	1	1	1	1	2	1	1	2	2	2	1	1	2	2	1	1	1	2	1	2
3	1	1	1	2	2	2	2	2	1	2	2	2	1	1	2	2	1	2	1	2	1	1	1	2	1	2	1	1	2	1	1	2	2	2	2	2
4	2	1	1	2	2	1	2	1	2	1	1	2	1	2	1	2	2	2	2	2	2	1	1	2	1	1	1	1	2	1	2	2	2	2	1	2
5	1	1	1	2	2	2	1	1	1	1	1	1	2	1	1	1	1	1	2	2	2	1	2	2	1	1	1	1	2	2	1	2	1	1	1	1
6	2	2	1	2	1	1	1	2	2	1	2	2	1	2	2	2	2	2	2	2	2	2	2	2	2	2	1	2	1	1	2	2	2	1	2	2
7	2	1	1	2	2	2	2	1	2	1	2	2	2	1	1	2	1	2	2	2	2	1	1	2	2	2	2	1	2	1	1	2	2	2	1	2
8	1	2	2	1	2	1	1	1	2	2	1	2	1	1	1	1	2	1	2	1	2	2	1	1	2	1	2	1	2	1	2	2	1	2	1	1
9	2	2	2	1	1	2	1	1	2	2	1	1	1	2	1	2	2	1	1	1	2	1	2	2	1	2	1	1	1	2	1	2	1	2	1	2
10	2	1	2	2	2	1	2	1	2	2	1	1	1	2	1	2	1	1	2	2	2	2	1	2	2	2	1	1	2	1	2	1	1	2	1	1
11	2	1	1	1	2	2	1	2	1	2	1	2	1	2	1	2	2	2	2	2	2	1	1	2	2	1	1	2	2	2	1	2	1	2	2	2
12	1	2	1	1	1	2	2	1	2	2	1	1	1	2	2	2	2	1	2	1	2	2	2	1	1	1	2	2	2	1	1	2	1	2	2	1
13	1	1	2	2	2	2	1	1	2	2	1	1	1	1	1	2	1	1	2	1	1	1	2	1	1	2	1	1	2	2	1	1	2	2	2	2
14	2	1	2	2	1	2	1	1	2	2	1	1	1	1	2	1	1	1	1	2	2	1	1	2	1	2	2	1	1	2	1	1	1	1	2	2
15	1	1	1	1	2	2	2	1	1	1	1	2	1	1	1	1	1	1	2	1	1	1	1	1	2	1	1	1	1	1	1	2	2	2	1	1

ID-NO	Q37	Q38	Q39	Q40	Q41	Q42	Q43	Q44	Q45	Q46	Q47	Q48	Q49	Q50	Q51	Q52	Q53	Q54	Q55	Q56	Q57	Q58	Q59	Q60	Q61	Q62	Q63	Q64	Q65	Q66	Q67	Q68	Q69	Q70
1	2	2	1	2	2	1	1	2	1	1	2	1	1	1	1	1	2	2	1	2	1	1	2	2	2	1	1	1	2	2	2	2	1	2
2	1	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	2	2	2	2	1	1	1	1	2	2	1	2	1	2	2	1	2
3	1	1	2	2	1	2	1	2	2	1	1	1	1	1	1	1	1	1	1	2	2	1	1	2	2	2	2	1	2	2	2	2	1	1
4	1	2	2	2	2	1	2	2	1	2	1	1	2	1	1	1	2	2	2	2	2	2	1	1	2	1	1	1	2	1	2	2	2	2
5	1	2	2	1	1	1	2	2	2	1	2	1	2	1	1	2	1	2	1	2	2	1	2	2	1	2	2	1	1	1	2	2	1	1
6	2	2	1	1	2	1	2	1	1	2	1	1	1	1	1	2	1	2	2	2	2	1	1	2	2	1	1	1	2	2	1	2	1	2
7	1	2	1	2	2	1	1	2	1	2	2	1	2	1	1	1	2	2	1	2	2	2	2	1	2	2	2	1	2	2	2	2	1	2
8	1	1	2	1	1	2	2	1	2	1	2	2	2	1	2	1	2	2	1	2	1	1	1	1	2	2	1	1	2	1	2	2	2	1
9	2	1	1	2	1	1	1	2	2	2	2	2	2	1	1	1	1	2	1	1	2	1	2	1	1	2	2	1	1	2	2	2	1	2
10	2	2	1	1	1	1	2	1	2	1	2	1	1	1	1	1	2	2	2	2	1	2	2	1	2	1	1	1	2	1	2	1	1	2
11	1	2	1	2	2	2	2	1	2	1	2	2	2	2	1	2	1	2	2	2	2	1	2	2	2	2	2	1	2	1	2	2	1	2
12	2	2	1	1	2	1	2	1	1	1	2	2	1	2	2	2	1	1	2	2	2	2	2	1	2	2	1	2	2	1	1	2	1	2
13	2	1	2	2	2	1	1	2	1	1	2	1	1	1	1	1	1	2	1	1	2	2	1	2	2	2	1	1	2	1	2	2	1	2
14	2	1	1	1	2	2	1	2	2	1	1	2	1	1	1	1	1	1	2	1	1	1	2	2	1	1	2	1	1	2	1	2	2	2
15	1	1	2	2	1	1	2	2	1	1	2	1	2	1	1	1	2	1	1	1	1	1	1	2	2	2	2	1	1	2	1	2	1	1

ID-NO	Q37	Q38	Q39	Q40	Q41	Q42	Q43	Q44	Q45	Q46	Q47	Q48	Q49	Q50	Q51	Q52	Q53	Q54	Q55	Q56	Q57	Q58	Q59	Q60	Q61	Q62	Q63	Q64	Q65	Q66	Q67	Q68	Q69	Q70
1	2	1	1	2	1	1	2	2	2	1	2	1	1	1	1	1	1	2	1	1	1	2	1	1	2	1	1	1	2	2	2	2	1	2
2	1	2	1	2	1	2	2	2	1	2	2	1	2	1	1	1	1	2	2	2	2	2	1	1	2	2	2	1	2	1	2	1	1	2
3	2	1	2	2	2	1	1	2	2	1	2	1	1	1	2	1	1	2	2	1	2	1	1	2	2	1	2	1	2	2	2	2	1	2
4	2	2	2	2	1	1	2	2	1	2	1	1	1	1	1	1	1	2	1	2	2	2	2	1	2	1	2	1	2	1	2	2	1	1
5	1	1	1	1	1	1	2	2	2	1	2	1	2	1	1	2	1	2	2	2	2	2	1	2	2	2	2	1	2	2	2	1	2	2
6	1	2	1	2	2	2	2	2	1	2	1	1	1	1	1	2	1	2	2	2	2	1	1	2	2	1	1	1	2	2	2	2	2	1
7	1	2	1	2	1	1	1	2	1	2	2	2	1	1	1	1	1	2	2	2	2	2	1	1	2	2	2	1	2	2	2	2	2	2
8	1	1	2	1	1	2	2	1	2	1	2	2	2	1	2	1	2	2	1	2	1	1	1	1	2	2	1	1	2	1	2	2	2	1
9	2	1	1	2	1	1	1	2	2	2	2	2	2	1	1	1	1	2	1	1	2	1	2	1	1	2	2	1	1	2	2	2	1	2
10	2	2	1	1	1	1	2	1	2	1	2	1	1	1	1	1	2	2	2	2	1	2	2	1	2	1	1	1	2	1	2	1	1	2
11	1	2	1	2	2	2	2	1	2	1	2	2	2	2	1	2	1	2	2	2	2	1	2	2	2	2	2	1	2	1	2	2	1	2
12	2	2	1	1	2	1	2	1	1	1	2	2	1	2	2	2	1	1	2	2	2	2	2	1	2	2	1	2	2	1	1	2	1	2
13	2	1	2	2	2	1	1	2	1	1	2	1	1	1	1	1	1	2	1	1	2	2	1	2	2	2	1	1	2	1	2	2	1	2
14	2	1	1	1	2	2	1	2	2	1	1	2	1	1	1	1	1	1	2	1	1	1	2	2	1	1	2	1	1	2	1	2	2	2
15	1	1	2	2	1	1	2	2	1	1	2	1	2	1	1	1	2	1	1	1	1	1	1	2	2	2	2	1	1	2	1	2	1	1

Appendix F

Summary of Interview Reinterview Table for All Questions

Question Number	**a**	**b**	**c**	**d**	κ
1	5	1	4	5	35.9
2	10	0	1	4	84.21
3	7	2	1	5	59.46
4	3	3	2	7	28.57
5	4	0	0	11	100
6	4	1	1	9	70
7	6	3	0	6	61.54
8	12	0	0	3	100
9	2	3	1	9	33.33
10	7	0	1	7	86.73
11	9	3	1	2	33.33
12	6	0	1	8	86.49
13	11	1	0	3	81.48
14	7	1	2	5	59.46
15	10	0	1	4	84.21
16	3	1	0	11	81.48
17	7	1	1	6	73.21
18	10	0	1	4	84.3
19	2	2	0	11	59.46
20	6	1	0	8	86.49
21	3	1	5	6	22.41
22	7	3	3	2	10
23	8	1	1	5	72.22
24	1	3	1	10	18.92
25	5	2	0	8	72.73
26	2	4	2	7	11.76
27	9	2	3	1	7.41
28	10	2	1	2	44.44
29	3	1	0	11	81.48
30	8	0	2	5	72.73
31	8	2	3	2	21.05
32	3	2	0	10	66.67
33	7	1	2	5	59.46
34	3	1	2	9	52.63
35	5	4	1	5	35.9
36	5	0	1	9	85.71
37	6	1	2	6	60.18
38	6	2	0	7	73.68
39	8	2	0	5	72.73
40	5	0	1	9	85.71
41	5	4	1	5	35.9
42	9	1	1	4	70
43	5	0	1	9	85.71
44	4	0	1	10	84.21
45	6	1	1	7	73.21
46	10	0	0	5	100
47	3	0	1	11	81.48
48	9	0	1	5	85.71
49	7	2	1	5	59.46
50	13	0	0	2	100
51	12	0	1	2	76.19
52	11	0	0	4	100
53	9	3	0	3	54.55
54	3	0	1	11	81.48
55	5	1	3	6	47.37
56	4	2	0	9	70.59
57	5	0	0	10	100
58	7	0	3	5	60.87
59	6	3	1	5	47.37
60	7	1	0	7	86.73
61	2	0	2	11	59.46
62	5	1	0	9	85.71
63	6	0	1	8	86.49
64	14	0	0	1	100
65	3	0	1	11	81.48
66	7	0	1	7	86.73
67	3	0	1	11	81.48
68	1	2	0	12	44.44
69	9	1	3	2	33.33
70	2	2	2	9	31.82

Appendix G

Avarage of Weights - Survey Data

ID NO	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12
1	0.75	0.50	0.50	0.50	0.75	0.50	0.50	0.50	0.50	0.50	0.75	0.75
2	0.75	0.50	0.75	1.00	0.75	0.50	0.50	0.75	0.25	0.50	1.00	0.75
3	0.75	0.25	0.50	0.50	0.75	0.50	0.75	0.75	0.25	0.50	0.75	0.50
4	0.75	0.50	0.50	0.50	0.50	0.25	0.75	0.75	0.50	0.50	0.50	0.50
5	0.75	0.75	0.50	0.50	0.75	0.75	0.50	0.50	0.50	0.50	0.50	0.50
6	1.00	0.75	1.00	1.00	0.75	1.00	1.00	1.00	0.75	0.75	1.00	0.50
7	0.75	0.50	0.75	0.75	0.75	0.75	0.50	0.75	0.50	0.50	0.50	0.75
8	0.75	0.50	0.75	0.50	0.75	0.50	0.75	0.75	0.50	0.50	1.00	0.50
9	0.75	0.50	0.75	0.50	0.75	0.50	0.50	1.00	0.50	0.50	0.75	0.50
10	0.75	0.50	0.50	0.50	0.75	0.50	0.50	0.75	0.50	0.75	0.75	0.75
11	0.75	0.75	0.50	0.50	0.75	0.25	0.75	0.50	0.50	0.50	1.00	0.75
12	0.75	0.75	0.75	0.75	0.75	0.25	0.75	0.75	0.50	0.50	0.50	0.75
13	0.75	0.50	0.50	0.50	0.75	0.75	0.50	0.75	0.50	0.50	0.50	0.75
14	0.75	0.50	0.50	0.50	0.75	0.50	0.75	0.75	0.50	0.75	0.75	0.75
15	0.75	0.50	0.50	0.75	0.75	0.75	0.75	0.75	0.50	0.50	0.75	0.75
16	1.00	0.75	0.75	0.50	0.75	0.75	0.75	0.75	0.50	0.50	0.75	0.75
17	0.75	0.50	0.75	0.50	0.75	0.75	0.50	0.50	0.50	0.50	0.50	0.50
18	0.75	0.75	0.75	0.50	1.00	0.50	0.75	0.50	0.50	0.50	0.75	0.75
19	0.75	0.75	1.00	1.00	0.75	0.75	0.50	1.00	0.25	0.50	0.75	0.75
20	0.75	0.25	0.75	0.75	0.75	0.50	0.75	0.75	0.50	1.00	0.50	0.50
21	0.75	0.75	0.75	0.75	0.75	0.50	0.50	1.00	0.50	0.75	0.75	0.75
22	1.00	0.75	0.75	0.75	1.00	0.25	0.75	0.75	0.50	0.50	0.75	0.75
23	0.75	0.50	0.50	0.50	1.00	0.50	0.75	0.50	0.50	0.50	0.75	0.75
24	1.00	0.50	0.50	0.50	0.75	0.50	0.50	0.75	0.50	1.00	1.00	0.75
25	0.75	1.00	0.25	0.25	1.00	0.75	0.25	1.00	0.25	1.00	0.25	0.25
26	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.50	0.50	0.75	0.75	0.75
27	1.00	0.50	0.75	0.50	1.00	0.50	0.75	1.00	0.25	0.50	0.50	1.00
28	0.50	0.25	0.75	0.75	0.50	0.50	0.50	1.00	0.25	0.50	0.75	0.50
29	0.75	0.50	0.50	0.50	1.00	0.75	0.75	1.00	0.50	0.75	0.75	0.50
30	0.75	0.50	0.75	0.50	0.75	0.50	0.75	0.75	0.50	0.75	0.75	0.75
31	1.00	0.75	1.00	0.75	0.75	0.50	0.75	0.50	0.75	0.75	0.50	0.75
32	0.75	0.50	0.75	0.75	1.00	0.50	0.50	1.00	0.50	0.75	0.75	0.75
33	0.75	0.50	0.75	0.50	1.00	0.50	0.50	0.50	0.25	0.50	0.75	0.75
34	1.00	0.75	1.00	1.00	0.75	0.25	0.75	0.25	0.50	0.50	0.75	0.50
35	0.75	1.00	1.00	0.50	0.75	0.25	0.75	0.50	0.25	0.75	0.75	1.00
36	1.00	0.50	0.75	0.75	1.00	0.50	1.00	0.75	1.00	0.75	1.00	0.75
37	0.75	0.25	0.75	0.50	0.75	0.50	0.50	0.75	0.75	0.75	0.75	0.50
38	0.75	0.75	0.75	0.75	0.50	0.50	0.75	0.75	0.50	0.75	0.75	0.75
39	0.75	0.50	1.00	0.75	1.00	0.50	0.50	0.75	0.75	0.50	0.75	1.00
40	0.75	0.50	0.50	0.50	1.00	0.25	0.75	0.50	0.75	0.75	0.75	0.75
41	0.75	0.50	0.75	0.75	0.75	0.50	0.75	1.00	0.50	0.50	1.00	0.50
42	0.75	0.25	0.75	0.25	0.50	0.75	0.50	0.75	0.25	0.50	0.50	0.25
43	1.00	0.75	0.50	0.25	1.00	0.50	0.50	1.00	0.25	0.75	1.00	0.50
44	1.00	0.50	1.00	1.00	0.75	0.25	0.75	0.75	0.25	0.50	1.00	0.75
45	0.50	0.75	0.75	0.75	0.75	0.50	0.75	0.75	0.25	0.50	0.75	0.75
46	0.75	0.50	0.75	0.50	0.75	0.75	0.50	0.50	0.25	0.25	0.75	0.75
47	0.75	0.75	0.75	0.50	1.00	0.50	1.00	0.25	0.50	0.50	1.00	1.00
48	0.75	0.75	0.75	0.75	0.75	0.50	0.75	0.75	0.75	0.75	0.75	0.25
49	0.75	0.25	1.00	0.75	0.75	0.75	0.50	0.75	0.25	0.75	0.50	0.75
50	0.75	1.00	0.50	0.75	1.00	0.75	0.50	1.00	0.50	1.00	1.00	0.25
51	0.75	0.25	0.75	0.25	0.25	1.00	0.75	0.75	0.50	0.75	0.25	0.75
52	0.75	0.50	0.75	0.75	1.00	0.75	0.75	0.75	0.50	1.00	0.75	0.75
53	1.00	1.00	1.00	1.00	1.00	0.25	1.00	0.50	1.00	1.00	1.00	0.75
54	0.75	0.75	0.50	0.50	0.75	0.75	0.50	0.75	0.50	0.50	0.75	0.25
55	0.75	1.00	0.75	0.75	0.75	0.75	0.75	0.50	0.50	1.00	0.75	0.25
56	1.00	0.75	1.00	0.75	1.00	0.75	0.50	0.75	0.50	1.00	1.00	0.75
57	0.75	0.75	0.75	0.50	1.00	0.75	0.75	0.50	0.50	0.50	0.75	0.25
58	0.75	0.75	0.75	0.75	0.75	0.50	0.75	0.50	0.50	0.50	0.75	0.50
59	1.00	0.25	1.00	0.75	0.75	0.50	0.75	1.00	0.25	0.75	0.75	0.75
60	0.75	0.75	0.75	0.50	0.75	1.00	0.75	0.75	0.50	1.00	0.75	0.50
61	1.00	0.50	1.00	1.00	0.75	0.25	0.75	0.75	0.50	0.50	1.00	0.50
62	0.75	0.50	0.75	0.75	1.00	0.50	0.75	0.50	0.50	0.75	0.75	0.75
63	1.00	0.75	1.00	1.00	0.75	1.00	0.50	1.00	0.25	1.00	1.00	0.75
64	0.75	0.75	0.75	0.75	0.75	0.50	0.50	0.75	0.50	0.75	0.50	0.50
65	0.75	0.75	0.50	0.75	0.75	0.50	0.75	0.50	0.50	0.75	0.75	0.50
66	0.50	1.00	0.50	1.00	0.75	0.25	0.75	1.00	0.75	1.00	1.00	0.75
67	1.00	0.25	1.00	0.75	0.75	0.25	0.75	1.00	0.50	0.75	0.75	0.75
68	0.75	0.50	0.75	0.75	0.75	0.25	0.50	0.75	0.75	0.50	0.75	0.50
69	1.00	0.25	0.75	0.25	1.00	1.00	0.75	0.75	0.25	0.50	1.00	0.25
70	0.50	0.75	0.50	0.75	1.00	0.25	0.50	1.00	0.25	0.50	0.75	0.75
71	0.75	0.75	0.75	1.00	0.75	0.25	1.00	1.00	0.50	0.75	1.00	0.25
72	1.00	1.00	0.25	1.00	0.75	1.00	0.50	1.00	0.25	0.25	1.00	0.50
73	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.50	0.75	0.75	0.50
74	0.75	0.75	0.75	0.75	0.75	0.50	0.75	0.50	0.50	0.50	0.75	0.50
75	1.00	0.50	0.50	0.50	0.75	0.75	0.50	1.00	0.75	0.75	0.75	1.00
76	1.00	1.00	1.00	1.00	0.75	0.75	0.75	0.50	0.50	0.75	0.75	0.75
77	1.00	0.75	1.00	0.75	0.50	0.50	0.75	0.75	0.25	0.75	0.75	0.75
78	0.75	0.25	0.75	0.50	0.75	0.25	0.50	0.75	0.50	1.00	0.75	0.25
79	0.75	0.75	0.75	0.75	0.75	0.50	0.75	0.50	0.50	0.50	0.75	0.50
80	0.75	0.25	0.50	0.75	0.50	0.75	0.75	1.00	0.50	1.00	0.75	0.50
81	0.75	0.25	1.00	0.25	0.75	0.75	0.75	0.50	0.25	0.50	0.75	0.50
82	0.75	0.50	0.75	0.75	0.50	0.50	0.75	0.75	0.50	0.50	0.50	0.25
83	0.75	0.50	0.75	0.75	0.75	0.50	0.75	0.75	0.50	0.50	0.75	0.50
84	0.50	0.50	0.75	0.75	0.75	0.50	0.75	0.75	0.75	0.75	0.75	0.25
85	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	1.00	0.50
86	1.00	0.25	1.00	1.00	1.00	0.25	1.00	0.75	0.50	1.00	1.00	1.00
87	1.00	0.50	1.00	0.75	0.75	0.25	0.75	0.75	0.75	0.75	0.75	0.75
88	0.75	0.50	0.75	0.50	0.75	0.75	0.25	0.50	0.25	0.50	0.75	0.75
89	0.75	0.50	0.50	0.50	1.00	0.50	1.00	0.75	0.75	0.75	0.75	0.75
90	0.75	0.50	0.50	0.75	0.75	0.50	0.75	0.75	0.75	0.50	0.75	0.50
91	0.75	0.50	1.00	0.75	0.75	0.50	0.75	0.75	0.50	0.75	0.75	0.75
92	1.00	0.75	1.00	0.50	1.00	1.00	1.00	1.00	0.25	1.00	1.00	0.25
93	0.75	0.75	1.00	0.75	0.75	1.00	0.75	0.75	0.50	0.50	0.75	0.75
94	0.75	0.50	0.75	0.75	0.75	0.50	0.75	0.75	0.50	0.50	0.75	0.75
95	0.50	0.75	0.50	0.50	0.75	0.75	0.50	0.75	0.75	0.75	0.75	0.50
96	0.75	0.25	0.75	0.75	0.75	0.50	0.50	0.75	0.50	0.75	0.75	0.75
97	0.75	0.50	0.75	0.75	0.75	0.50	0.75	0.75	0.75	0.50	0.50	0.75
98	0.75	0.25	1.00	0.75	0.75	0.25	1.00	0.75	0.50	0.50	1.00	0.50

ID NO	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12
99	0.75	0.75	0.50	0.50	0.75	0.75	0.50	0.75	0.75	0.75	0.75	0.25
100	1.00	0.75	0.75	0.50	0.75	0.75	0.50	0.50	0.25	0.50	0.75	0.50
101	0.75	0.75	0.75	0.50	1.00	0.50	0.75	1.00	0.25	0.75	0.75	0.75
102	0.75	0.50	0.75	0.75	0.75	0.50	0.75	0.75	0.50	1.00	0.75	0.25
103	0.75	0.50	0.75	0.50	0.75	0.50	0.75	0.75	0.50	0.50	0.75	0.25
104	0.75	0.75	0.75	0.75	0.75	0.50	0.75	0.75	0.75	0.75	0.75	0.50
105	0.75	0.75	1.00	0.75	0.75	0.75	0.50	0.50	0.25	0.75	0.75	0.25
106	0.75	0.50	0.75	0.50	1.00	0.25	1.00	0.50	0.50	0.50	0.75	0.50
107	0.75	0.25	1.00	1.00	1.00	0.25	0.75	0.75	0.25	0.75	0.75	0.25
108	1.00	0.75	0.75	1.00	0.75	0.25	1.00	1.00	0.25	1.00	1.00	0.75
109	1.00	0.25	1.00	0.75	1.00	0.25	1.00	1.00	1.00	1.00	1.00	0.75
110	0.75	0.50	0.75	0.50	0.75	0.75	0.50	1.00	0.75	0.75	1.00	0.50
111	0.75	0.50	0.75	0.50	0.75	0.50	0.75	0.50	0.50	0.75	0.75	0.25
112	0.75	0.75	0.50	0.50	0.75	0.50	0.75	0.75	0.50	0.50	0.50	0.50
113	0.75	0.75	0.75	0.75	0.50	0.25	0.75	0.75	0.25	0.50	0.75	0.75
114	0.75	0.75	0.50	0.50	0.75	0.75	0.50	0.75	0.50	0.50	0.50	0.75
115	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75
116	0.75	0.75	0.50	0.50	0.75	0.75	0.50	0.75	0.50	0.75	1.00	0.75
117	0.75	0.50	0.75	0.75	0.75	0.75	0.50	0.75	0.75	0.75	0.75	0.75
118	1.00	1.00	0.75	1.00	1.00	0.25	1.00	1.00	1.00	1.00	1.00	0.25
119	0.75	0.25	0.75	0.50	1.00	0.50	0.50	0.75	0.50	0.25	0.50	0.50
120	0.75	0.50	0.75	0.50	0.75	0.50	0.75	0.50	0.50	0.50	0.75	0.50
121	0.75	0.75	0.50	0.50	0.75	0.50	0.75	1.00	0.50	0.75	1.00	0.50
122	0.75	0.50	0.50	0.50	0.75	0.50	0.75	0.75	0.75	0.50	0.75	0.75
123	0.75	1.00	0.75	0.75	0.50	0.75	0.75	1.00	1.00	0.50	1.00	0.75
124	0.75	0.50	0.50	0.50	0.75	0.25	0.75	0.50	0.50	0.50	1.00	0.75
125	1.00	0.25	0.25	1.00	0.75	1.00	0.50	1.00	0.25	0.25	0.50	0.25
126	0.75	0.50	1.00	1.00	0.75	1.00	0.75	0.75	0.50	0.75	0.75	0.50
127	1.00	1.00	1.00	1.00	1.00	0.50	1.00	0.75	0.25	0.75	1.00	0.50
128	0.75	0.75	0.75	0.75	0.75	0.50	0.75	0.75	0.50	0.75	0.75	0.50
129	1.00	0.75	0.75	0.50	0.75	0.25	0.50	1.00	0.50	1.00	0.75	0.50
130	1.00	0.50	0.75	0.50	0.75	0.25	1.00	0.75	0.75	0.75	0.50	0.75
131	0.75	0.50	0.75	0.75	0.75	0.75	0.75	0.50	0.25	0.50	0.50	1.00
132	1.00	0.75	1.00	0.50	0.75	0.50	0.75	0.75	0.25	0.75	0.50	0.75
133	0.75	0.50	0.50	0.75	0.75	0.75	0.50	0.50	0.50	0.50	0.75	0.50
134	1.00	0.50	0.75	0.75	0.75	0.75	0.75	0.50	0.25	0.75	0.75	0.75
135	0.75	0.25	1.00	0.75	1.00	0.25	1.00	1.00	1.00	0.75	0.75	0.25
136	1.00	0.50	1.00	0.75	0.75	0.25	0.75	0.75	0.25	0.75	0.75	0.75
137	1.00	1.00	1.00	0.75	0.75	0.25	0.75	0.50	0.75	0.50	0.75	0.25
138	0.75	0.75	1.00	0.75	0.75	0.50	0.75	0.50	0.50	0.50	0.75	0.25
139	0.75	0.75	0.75	1.00	0.75	0.50	0.75	0.50	0.50	1.00	0.75	0.75
140	0.75	1.00	0.75	1.00	0.75	0.50	0.75	1.00	1.00	0.50	1.00	0.75
141	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75
142	0.75	0.50	0.75	0.50	0.75	0.50	0.50	0.50	0.75	0.75	0.50	0.75
143	0.75	0.50	0.75	0.75	0.75	0.50	0.75	1.00	0.25	0.75	0.75	0.25
144	0.75	0.50	0.75	0.75	0.75	0.50	0.75	0.25	0.50	0.50	0.75	0.25
145	0.50	0.75	0.75	0.50	0.75	0.75	0.75	0.75	0.50	0.75	0.50	0.75
146	0.75	0.50	0.75	0.75	0.75	0.50	0.50	0.75	0.50	0.50	0.75	0.75
147	0.75	0.50	0.75	0.50	0.50	0.75	0.50	0.75	0.25	0.25	0.75	0.75
148	0.50	0.75	0.50	0.50	0.75	0.75	0.25	0.50	0.25	0.50	0.75	0.75
149	0.50	0.25	0.75	0.75	0.75	0.25	0.50	1.00	0.50	0.50	0.75	0.75
150	0.75	0.50	0.75	0.50	1.00	0.25	1.00	1.00	0.25	1.00	1.00	0.50
151	0.75	0.75	0.50	0.75	0.75	0.25	0.50	0.75	0.75	0.50	0.75	0.75
152	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.25	0.75	0.75	0.50
153	1.00	0.50	0.75	0.50	0.75	0.25	0.50	0.75	0.50	1.00	0.50	0.75
154	1.00	0.25	0.50	0.75	0.75	0.25	1.00	1.00	0.75	0.50	1.00	0.75
155	1.00	0.75	0.75	0.50	0.50	0.50	0.50	1.00	0.25	0.50	0.75	0.75
156	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.50
157	0.75	0.75	1.00	1.00	1.00	0.25	0.75	1.00	0.50	1.00	0.75	0.50
158	0.75	0.50	0.75	0.75	0.50	0.50	0.75	1.00	0.25	1.00	1.00	0.25
159	0.75	0.25	0.75	0.25	0.50	0.75	0.25	0.75	0.25	0.75	0.75	0.25
160	1.00	1.00	1.00	0.75	1.00	0.25	1.00	0.25	0.75	1.00	1.00	0.75
161	0.75	0.75	0.75	0.25	0.75	0.50	0.75	0.50	0.50	0.75	0.75	0.50
162	1.00	1.00	1.00	1.00	1.00	1.00	0.75	1.00	0.50	1.00	1.00	1.00
163	1.00	1.00	0.75	0.50	1.00	0.50	1.00	1.00	0.75	0.75	1.00	0.50
164	0.75	0.75	0.75	0.50	0.50	0.50	0.50	1.00	0.50	0.50	0.75	0.50
165	0.75	0.50	0.75	0.50	0.75	0.75	0.75	0.75	0.50	0.75	0.75	0.75
166	0.50	0.75	0.75	0.50	0.75	0.75	0.75	0.75	0.50	0.75	0.75	0.75
167	0.75	0.75	0.75	0.50	1.00	0.50	0.75	1.00	0.50	0.75	0.75	0.75
168	0.75	0.50	0.75	0.75	0.75	0.25	0.75	0.75	0.50	0.75	0.75	0.50
169	0.75	0.50	0.75	0.75	0.75	0.50	0.50	0.75	0.75	0.75	0.75	0.50
170	0.75	0.75	0.75	0.75	0.75	0.75	0.50	1.00	0.75	0.75	0.75	0.75
171	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75
172	0.75	0.50	0.75	0.75	1.00	0.50	0.75	0.75	0.50	0.50	1.00	0.50
173	0.75	0.50	0.50	0.75	0.75	0.75	0.50	1.00	0.50	0.50	0.75	0.75
174	0.75	0.25	1.00	0.25	1.00	0.75	0.50	0.75	0.75	0.75	1.00	0.50
175	0.75	0.50	0.50	0.75	0.75	0.50	0.75	0.75	0.50	0.50	0.75	0.75
176	0.75	0.75	0.75	0.50	1.00	0.50	0.50	0.75	0.50	0.75	1.00	0.50
177	0.75	0.50	0.50	0.25	0.75	0.75	0.75	0.50	0.50	0.75	0.75	0.50
178	1.00	1.00	1.00	0.75	1.00	0.25	1.00	0.50	1.00	1.00	1.00	0.75
179	0.75	0.50	0.75	0.75	0.75	0.25	0.75	0.50	0.50	0.50	0.75	0.50
180	0.75	0.50	0.50	0.25	0.50	0.50	0.75	0.50	0.50	0.50	0.50	0.25
181	0.75	0.50	0.50	0.75	0.75	0.25	0.50	0.50	0.50	0.50	0.75	0.50
182	0.75	0.50	0.75	0.75	0.75	0.50	0.75	0.75	0.50	0.75	0.50	0.75
183	1.00	0.75	0.75	0.75	0.75	0.50	0.75	0.50	0.50	1.00	0.75	0.75
184	1.00	0.50	1.00	0.75	0.75	0.50	0.50	1.00	1.00	0.50	1.00	1.00
185	0.75	0.50	0.50	0.25	0.75	0.75	0.75	0.75	0.50	0.75	0.75	0.50
186	0.75	0.50	0.75	0.50	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75
187	1.00	0.75	0.50	0.50	1.00	0.25	1.00	0.75	0.25	1.00	1.00	0.25
188	0.75	0.50	0.75	0.50	0.75	0.50	0.50	0.75	0.50	0.50	0.75	0.50
189	0.75	0.50	0.75	0.50	0.75	0.50	0.50	1.00	0.25	0.75	0.50	0.75
190	0.75	1.00	1.00	0.50	1.00	0.75	1.00	1.00	0.75	1.00	1.00	0.50
191	0.75	0.50	0.75	0.50	0.75	0.50	0.75	0.50	0.50	0.50	0.75	0.75
192	0.75	0.75	0.75	0.50	0.75	0.50	1.00	1.00	0.50	0.75	1.00	0.75
Average of Weights	0.80	0.61	0.74	0.65	0.79	0.55	0.69	0.75	0.51	0.68	0.77	0.60

Appendix H

Industrial Implementation of Card Game Data

ID-NO	SEX	JOB TITLE	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15	Q16	Q17	Q18	Q19	Q20	Q21	Q22	Q23	Q24	Q25	Q26	Q27	Q28	Q29	Q30	Q31	Q32	Q33	Q34	Q35	Q36	Q37	Q38	Q39	Q40	Q41	Q42	Q43	Q44	Q45
1	M	Software developer	1	1	1	2	1	2	1	1	1	2	1	1	1	2	2	1	2	2	2	1	2	1	2	1	2	1	2	1	2	1	1	2	1	1	1	2	1	2	1	2	1	2	2
2	M	Software developer	1	2	1	2	2	2	1	2	2	2	1	1	1	2	1	1	2	2	2	2	2	1	1	2	2	2	1	2	2	1	2	2	1	2	2	1	1	1	2	2	1	2	1	1	2
3	M	Software specialist	1	2	1	2	2	2	1	1	2	2	2	1	1	1	2	1	1	1	1	2	2	1	1	2	1	2	1	1	1	1	2	1	2	2	1	2	1	1	2	2	1	1	2	2	2
4	M	Software specialist	1	2	1	2	2	2	2	2	2	1	1	1	2	1	2	2	2	2	2	1	2	1	2	2	2	2	1	1	2	1	2	1	2	2	1	1	1	1	2	1	1	1	2	2	2
5	M	Software specialist	1	2	1	2	2	1	1	1	2	1	1	2	2	2	1	2	1	1	2	2	2	1	1	2	1	2	1	1	1	1	2	2	2	1	2	2	1	2	2	1	1	1	2	2	2
6	F	Software Tester	1	2	1	2	2	2	2	1	2	2	1	1	2	2	2	2	1	1	2	1	2	1	2	2	1	2	1	1	1	2	2	1	1	1	1	1	2	2	1	1	1	2	2	2	1
7	M	Software specialist	1	2	2	2	2	2	2	1	1	1	2	1	2	2	1	2	2	2	1	1	2	1	1	2	1	2	1	1	1	2	2	1	1	1	2	2	2	2	1	1	1	1	2	2	2
8	M	Software specialist	2	1	2	2	2	1	2	1	2	1	1	2	1	1	1	2	2	2	1	2	2	1	1	1	2	2	1	1	2	2	1	2	2	1	2	2	1	1	1	2	1	1	2	2	1
9	F	Software developer	1	2	1	2	2	1	2	1	2	2	1	1	1	1	2	2	1	2	2	2	2	1	2	2	1	2	1	1	1	2	2	2	1	2	1	1	2	1	2	2	1	2	2	1	1
10	F	Software developer	1	2	2	1	2	2	1	1	2	1	1	2	2	1	1	2	1	1	1	2	2	1	1	2	2	2	1	2	1	1	2	1	2	2	2	2	1	1	2	2	2	1	1	2	2
11	F	System Analyst	1	1	2	2	1	2	2	1	2	2	1	1	1	1	2	2	2	2	2	2	2	1	1	2	2	2	1	1	2	2	1	1	2	2	2	2	2	2	2	1	2	2	1	2	1
12	F	System Analyst	2	2	1	2	2	2	2	1	2	2	1	2	1	1	2	2	1	2	2	1	2	1	1	2	2	2	1	2	1	1	1	2	2	1	1	2	1	1	1	2	2	2	1	2	1
13	M	Software developer	2	2	1	2	1	2	2	2	2	2	1	2	1	2	1	2	2	1	2	2	2	2	1	1	2	1	2	1	2	2	1	1	2	2	1	1	1	1	2	2	2	1	1	2	2
14	M	Software developer	1	2	2	1	1	2	2	2	2	2	1	1	2	1	2	2	2	1	1	2	2	2	1	2	1	2	1	1	1	1	2	2	2	2	2	2	2	2	2	2	2	2	1	2	2
15	M	Project manager	1	1	2	2	2	1	2	1	2	2	1	1	1	1	2	2	1	2	2	1	1	1	1	1	1	2	1	1	2	2	2	2	2	2	2	2	1	2	1	1	1	1	1	2	2
16	M	Software developer	1	2	1	2	2	2	2	1	2	2	1	1	1	1	2	2	1	2	2	2	2	1	1	2	1	2	1	1	1	2	2	1	1	1	1	2	2	1	1	1	1	2	2	1	2
17	F	Software specialist	2	1	2	2	2	2	2	1	2	2	1	2	1	2	1	2	1	2	1	2	1	1	2	2	2	2	1	1	1	1	1	1	2	1	1	1	1	1	2	1	1	1	1	1	1
18	M	Software developer	1	2	2	1	2	2	2	1	1	2	1	2	2	1	1	1	1	2	2	2	2	1	1	2	1	2	1	1	1	1	2	2	2	2	1	2	1	2	1	1	1	2	1	2	2
19	M	Software specialist	1	2	2	2	2	2	1	2	1	2	1	1	1	2	2	2	2	1	1	2	2	2	2	2	2	2	1	1	2	2	2	2	2	2	2	2	2	2	1	2	2	1	2	1	2
20	M	Software developer	1	2	2	2	2	2	2	1	2	2	1	2	2	2	2	2	2	2	1	2	2	1	2	2	1	2	1	1	2	2	2	2	1	1	2	2	2	2	1	2	1	1	2	2	2
21	M	Software developer	1	2	1	2	2	2	2	2	1	1	2	2	2	2	2	2	2	1	1	1	2	2	2	1	1	2	1	1	1	2	1	1	2	2	2	1	2	1	2	1	2	1	1	2	2
22	M	Software developer	1	1	2	2	1	2	2	1	2	2	2	1	2	2	2	2	1	1	2	1	2	1	2	2	2	1	2	1	1	1	1	1	2	1	1	2	2	2	2	2	2	2	1	1	2
23	M	Software Tester	1	2	2	1	2	2	2	1	2	2	1	1	1	2	2	2	1	1	2	1	1	2	1	2	2	1	2	1	1	1	1	2	1	1	2	2	2	1	2	2	2	1	1	2	1
24	M	Software Tester	2	1	1	2	2	2	2	1	2	2	1	1	2	2	1	1	1	1	2	2	2	1	1	2	1	2	1	1	1	1	2	2	2	2	2	1	1	2	2	1	1	2	2	2	2
25	F	Software developer	1	2	2	2	2	1	2	1	1	1	1	1	1	2	1	2	2	1	2	1	2	1	1	2	1	2	1	2	1	1	2	2	2	1	2	2	2	2	1	1	2	2	2	1	1
26	F	Software developer	2	2	2	2	2	1	2	1	2	1	1	1	2	2	1	2	2	2	2	1	2	1	1	2	1	2	1	1	2	2	2	1	2	2	2	2	1	1	2	1	1	2	2	2	1
27	M	Software specialist	2	2	2	2	2	2	2	1	2	2	1	2	1	2	2	2	1	2	2	1	2	1	1	1	1	2	1	1	1	1	2	2	2	2	2	2	2	2	1	2	1	1	2	2	2
28	F	Software specialist	1	2	1	2	2	2	2	1	2	2	1	1	1	1	2	2	2	1	1	1	2	1	2	2	1	2	1	1	1	2	2	2	2	1	1	2	1	2	2	2	2	1	1	2	2
29	F	System Analyst	2	1	1	2	2	2	2	1	2	1	1	2	2	1	1	2	2	2	2	2	2	1	1	2	2	2	1	1	2	2	2	2	1	1	2	1	1	2	2	2	2	1	2	2	2
30	F	System Analyst	1	1	1	2	2	1	2	1	2	2	1	2	2	2	2	2	2	1	1	1	2	1	1	2	1	2	1	1	2	1	2	2	2	1	1	2	2	2	1	2	2	1	2	1	2
31	M	Software specialist	2	2	1	2	2	2	2	2	2	2	1	1	2	1	1	2	1	2	2	1	2	1	1	2	1	2	1	2	1	1	2	2	2	2	1	2	1	1	2	2	1	2	1	1	2
32	M	Software specialist	1	2	1	2	2	2	1	1	2	2	1	1	1	2	2	2	1	1	2	2	2	1	1	2	1	2	1	2	1	2	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1
33	M	Project manager	1	1	2	2	2	2	2	1	1	1	2	2	2	1	1	1	1	2	1	1	1	1	1	2	1	2	2	1	2	1	2	2	1	1	1	2	2	2	2	2	1	2	2	1	1
34	M	Software developer	1	1	2	1	2	2	2	1	2	2	1	1	2	1	2	2	1	1	2	2	2	1	1	2	1	2	1	1	1	1	1	2	2	2	2	2	1	1	2	1	1	1	1	2	2
35	F	Software Tester	2	1	1	2	2	2	2	1	2	2	1	1	2	1	1	2	1	2	2	1	2	1	2	2	1	2	2	1	2	1	2	2	2	1	2	2	1	1	2	2	1	1	1	2	2
36	M	Software specialist	2	2	2	2	2	1	2	1	1	2	1	1	1	1	1	2	1	1	2	2	2	1	2	2	1	2	1	1	1	2	2	2	1	1	2	1	1	2	2	2	1	1	1	2	2
37	M	Software Tester	1	1	1	2	2	2	2	1	2	2	1	1	1	2	1	2	2	1	2	2	1	1	1	2	1	2	1	1	1	2	2	2	2	1	2	1	1	2	2	2	1	1	1	1	2
38	M	Software developer	2	2	2	1	1	1	2	2	2	2	1	1	1	2	2	1	2	1	2	2	2	1	1	2	1	2	1	1	1	1	2	2	1	2	1	2	2	2	2	1	1	2	2	2	2
39	F	Software Tester	1	2	2	2	2	1	2	1	2	1	1	2	1	1	1	2	1	2	1	1	1	1	1	2	1	2	1	2	2	1	2	1	2	1	1	1	1	2	1	2	1	1	2	2	1
40	F	Software developer	2	2	2	2	1	2	2	2	1	1	1	2	1	2	2	1	2	2	2	2	2	2	2	1	2	2	1	1	2	2	2	2	1	2	2	2	1	2	2	1	2	2	1	1	2
41	M	Project manager	1	1	1	2	2	2	1	2	2	2	2	1	2	1	2	2	1	2	2	1	2	1	1	2	2	2	2	1	2	2	2	1	1	2	1	2	1	2	2	2	2	1	2	2	2
42	M	Software developer	2	1	1	2	2	2	2	1	2	2	1	1	1	2	2	2	2	2	2	2	2	2	1	2	2	2	2	2	1	1	2	2	1	2	2	2	2	1	2	1	2	1	1	1	2
43	M	Software specialist	1	1	2	1	1	2	2	1	2	2	1	1	2	1	2	1	1	2	1	1	2	1	1	2	2	1	2	1	1	1	2	2	2	1	2	2	1	2	1	1	1	1	1	2	2
44	M	Software Tester	1	2	2	2	2	1	2	1	1	1	1	1	1	2	1	2	1	1	1	1	1	1	2	2	1	1	1	1	1	2	2	2	2	1	1	1	2	1	2	2	1	2	1	2	2
45	M	Software developer	2	2	1	2	2	2	2	1	2	2	1	1	2	1	1	2	1	2	2	1	2	1	1	2	1	2	1	1	1	2	2	2	1	2	1	2	1	1	1	1	2	2	2	2	2
46	M	Software developer	1	2	2	2	2	2	1	1	2	2	1	2	2	2	1	2	2	1	2	2	1	1	2	2	2	1	1	1	1	2	2	1	2	2	2	1	2	2	2	1	1	2	2	2	1
47	M	Software developer	1	2	2	2	2	1	1	1	2	1	1	1	1	1	2	2	1	2	2	2	2	1	1	2	1	2	1	1	2	1	2	2	1	1	2	2	1	2	2	2	1	1	2	2	1
48	F	Software specialist	2	2	2	2	2	2	2	2	2	2	1	2	2	1	2	2	2	2	2	2	1	1	2	1	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	1	2	2	2	2
49	F	Software specialist	2	2	1	2	2	1	2	1	2	2	1	1	2	2	1	2	2	2	2	2	1	2	1	2	2	1	2	1	1	1	2	2	2	2	1	1	2	1	1	2	2	1	1	1	2
50	M	Software developer	2	2	2	2	2	2	2	1	2	1	1	2	2	2	2	2	2	2	2	2	2	1	1	2	1	2	1	1	2	2	2	1	1	2	2	2	1	2	1	2	2	2	1	1	2
51	M	Software Tester	1	1	2	2	2	2	2	1	1	2	1	1	1	2	2	2	2	2	2	1	2	1	1	1	2	2	1	2	2	2	2	2	1	2	1	2	1	2	2	2	2	1	2	2	2
52	M	Software specialist	1	2	1	2	2	2	1	2	1	1	1	2	2	1	2	2	2	2	2	1	1	1	1	2	2	2	2	1	1	1	2	2	2	1	1	1	1	2	2	2	1	2	1	2	1
53	M	Software developer	1	2	1	2	2	1	2	1	1	2	1	1	2	2	1	2	2	2	2	2	2	1	1	2	1	2	1	1	1	2	2	2	2	2	2	2	1	1	2	2	1	2	1	2	1
54	M	Software specialist	2	2	2	2	2	2	2	1	2	1	2	1	2	1	1	2	2	2	1	2	2	1	2	2	1	2	1	1	2	1	2	2	2	2	2	2	2	2	1	1	2	1	2	1	1
55	F	Project manager	1	1	1	2	2	2	2	1	1	2	1	1	2	2	1	2	2	2	1	2	2	1	2	2	1	2	1	1	1	1	2	2	2	1	1	1	2	1	1	2	1	2	1	2	2
56	F	Project manager	2	2	2	2	2	1	2	1	1	2	1	1	2	2	2	1	1	1	2	2	2	2	1	2	1	2	1	1	2	2	2	2	2	1	1	2	1	1	2	1	1	2	2	2	1
57	M	Project manager	1	1	1	2	1	1	1	1	2	1	2	1	2	2	1	2	1	2	1	2	2	1	1	2	1	2	1	1	1	1	2	2	1	2	1	1	1	2	2	2	2	2	1	2	1
58	M	Software specialist	1	1	2	2	1	2	2	2	2	1	2	2	2	1	2	1	1	2	2	2	2	1	1	2	2	2	1	1	2	1	2	2	1	1	2	1	1	2	2	2	2	1	2	2	2
59	M	Software developer	1	1	2	2	2	2	2	1	1	2	1	2	2	2	2	2	2	2	2	2	2	1	1	1	2	2	2	1	1	2	1	2	2	2	2	2	2	2	2	1	1	2	2	2	2
60	F	Software specialist	1	2	2	2	2	1	2	1	1	2	1	1	2	2	2	2	2	2	2	2	2	2	1	2	2	2	2	1	1	2	1	1	2	1	2	1	2	1	1	1	1	1	1	2	1
61	F	Software developer	1	2	2	2	2	2	2	2	2	1	1	1	2	2	2	1	1	2	2	2	2	2	2	1	1	2	2	2	1	2	2	2	2	1	2	2	2	2	1	1	1	1	1	2	2
62	M	Project manager	1	1	1	2	2	2	2	2	1	2	2	2	2	2	2	1	2	2	1	2	2	2	2	1	1	2	2	2	1	1	2	2	2	1	2	2	1	1	1	1	1	1	1	1	2
63	M	Software specialist	2	2	2	2	2	2	2	2	1	1	1	2	2	2	2	2	1	2	2	2	2	2	2	1	2	2	2	2	1	2	1	1	2	2	2	2	2	2	2	2	2	1	1	2	2	2	2

ID-NO	Q46	Q47	Q48	Q49	Q50	Q51	Q52	Q53	Q54	Q55	Q56	Q57	Q58	Q59	Q60	Q61	Q62	Q63	Q64	Q65	Q66	Q67	Q68	Q69	Q70
1	1	2	2	1	1	1	1	1	1	2	2	1	1	2	1	2	2	2	1	1	2	2	2	2	2
2	2	2	2	2	1	1	1	2	2	1	2	1	1	2	1	2	2	2	1	1	2	2	2	2	2
3	2	2	1	2	1	1	1	2	1	1	2	1	1	2	1	2	2	1	1	1	2	2	1	1	2
4	2	2	2	1	2	2	1	1	1	2	1	2	1	1	1	1	2	2	1	2	2	1	2	1	2
5	1	2	1	2	1	2	1	2	2	2	2	2	1	1	1	2	1	2	1	2	1	2	2	1	1
6	1	2	1	2	1	1	1	1	2	1	1	1	1	2	2	2	1	1	1	2	2	2	1	1	1
7	2	1	2	2	1	1	1	1	1	2	2	1	1	2	2	2	1	1	2	2	2	2	1	2	2
8	1	2	1	2	1	1	1	1	2	1	2	1	2	1	1	2	1	2	1	2	2	1	2	2	2
9	2	2	1	2	1	1	1	2	2	2	1	2	1	1	1	2	1	1	1	2	2	1	1	2	2
10	2	1	1	1	2	2	1	1	2	2	2	1	1	2	2	2	1	2	1	2	2	2	1	1	2
11	1	1	1	2	1	1	1	1	2	2	2	2	2	1	1	1	2	2	1	2	1	2	2	2	2
12	1	1	2	1	1	1	1	2	1	1	1	1	2	2	1	2	2	1	1	2	1	2	2	2	1
13	2	2	1	2	1	2	1	1	2	2	2	1	2	2	2	1	2	2	1	2	2	2	1	2	2
14	1	2	1	2	2	1	2	1	2	2	2	2	2	2	1	2	2	2	2	1	2	1	2	2	2
15	1	2	2	1	1	2	1	1	2	1	1	1	2	2	2	2	1	1	2	2	2	2	2	1	2
16	2	1	1	2	1	1	2	2	2	1	2	1	1	2	1	2	2	1	1	1	2	2	1	2	1
17	1	1	1	2	1	1	1	1	2	2	1	1	1	2	1	1	1	1	2	2	1	2	2	2	2
18	2	2	2	2	1	1	2	1	2	2	1	1	1	1	2	2	1	2	1	1	2	1	2	2	2
19	1	1	2	1	2	2	1	1	2	1	2	2	2	1	1	2	2	2	1	2	1	2	2	2	2
20	2	2	2	2	1	2	1	1	2	2	2	2	2	1	2	2	2	2	2	2	2	2	2	1	1
21	2	2	2	2	2	2	1	2	1	2	2	2	2	1	1	2	2	2	2	2	2	2	2	2	1
22	1	2	1	2	2	2	1	1	2	1	1	1	2	2	2	2	2	1	2	1	2	2	2	1	2
23	2	1	1	1	1	1	1	1	2	1	1	1	2	2	1	2	2	1	1	2	2	2	2	1	1
24	2	2	1	1	1	1	1	1	2	2	2	2	2	2	1	2	2	2	2	1	2	2	2	1	2
25	1	2	1	1	1	1	1	1	2	1	2	1	2	2	2	2	1	2	1	2	2	2	2	1	1
26	1	1	2	2	1	1	1	1	2	1	2	2	1	2	2	1	1	2	1	2	2	1	2	2	1
27	2	2	1	2	2	1	1	2	2	2	1	2	1	2	1	1	2	1	1	2	2	2	2	2	1
28	1	1	1	2	1	1	1	2	2	1	2	2	2	1	1	1	1	2	2	2	2	1	1	2	1
29	1	2	1	2	1	1	1	1	2	2	1	1	2	2	1	2	2	1	1	2	2	2	2	1	1
30	2	2	1	2	1	1	1	1	2	2	1	2	1	2	2	2	2	1	1	2	2	2	1	1	1
31	1	2	1	2	1	1	2	1	2	2	2	1	2	1	1	2	2	2	1	2	2	2	1	1	2
32	1	2	1	2	1	2	1	1	2	2	1	1	1	2	1	1	2	2	1	1	2	2	1	1	2
33	1	2	1	2	1	2	1	1	1	2	2	2	1	2	2	2	1	2	2	2	2	2	2	1	1
34	1	2	2	1	1	1	2	1	2	2	1	1	2	1	2	2	2	2	1	2	2	1	1	1	2
35	1	1	2	2	2	1	2	1	2	2	1	1	1	2	2	2	2	1	1	2	2	2	1	1	1
36	2	2	1	2	1	2	1	1	2	1	2	1	2	2	1	2	1	2	1	2	1	2	2	2	2
37	2	1	1	2	1	2	1	1	2	1	2	1	2	1	2	1	2	1	1	1	2	1	1	1	2
38	2	1	1	2	2	1	1	1	2	2	2	1	2	2	2	2	2	1	1	2	2	2	2	2	2
39	2	1	1	2	1	2	1	1	2	1	1	2	2	1	1	2	1	1	1	2	2	2	1	2	1
40	2	2	1	2	2	2	1	1	2	1	2	2	1	2	2	2	1	1	1	2	2	1	2	2	2
41	1	2	2	2	2	1	1	1	2	2	2	2	2	2	2	2	2	1	2	1	2	2	2	1	1
42	1	1	1	2	1	1	1	1	1	1	2	1	1	1	2	2	2	1	2	1	2	2	2	1	1
43	1	1	1	1	2	1	1	1	2	2	1	2	2	1	2	1	2	1	1	2	1	2	1	1	2
44	1	1	2	2	1	2	1	1	2	1	2	1	2	2	1	2	1	1	1	2	2	1	2	1	2
45	1	2	2	2	2	1	1	1	2	2	1	1	1	2	2	2	1	2	1	2	2	2	2	2	1
46	2	2	1	2	1	1	2	1	1	2	2	1	2	1	2	2	2	2	2	2	1	2	2	1	2
47	1	1	1	2	1	1	1	1	2	1	2	2	1	1	1	2	1	1	2	1	1	2	1	2	2
48	2	2	1	2	1	1	2	2	2	1	2	2	2	2	2	2	1	2	2	2	2	2	2	2	2
49	1	1	1	2	1	1	2	2	2	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1	2
50	2	2	2	1	1	1	2	2	2	2	2	1	1	2	2	2	2	2	1	2	2	2	2	2	2
51	2	2	1	2	2	1	1	1	2	2	2	2	2	2	1	2	1	2	1	2	1	1	2	1	2
52	2	2	2	2	1	1	1	1	2	1	1	1	1	1	1	2	2	2	1	2	2	2	2	2	1
53	2	2	1	2	1	2	1	1	2	1	1	1	1	1	2	2	2	2	1	2	2	2	2	1	2
54	1	2	2	2	1	2	2	1	2	2	2	1	2	2	2	2	2	1	1	2	2	2	1	1	2
55	1	1	2	2	1	2	1	1	2	2	1	1	2	1	2	2	2	1	1	1	2	2	2	2	2
56	1	2	2	1	1	2	1	1	2	2	1	1	1	2	2	2	2	2	2	2	2	2	2	1	1
57	1	1	2	2	1	1	1	1	2	2	2	1	1	2	2	2	1	2	1	2	1	2	1	1	1
58	1	1	2	2	1	1	1	1	1	1	1	1	1	2	2	2	2	1	2	2	2	2	1	2	1
59	2	2	1	2	1	2	1	1	2	2	2	1	1	2	2	2	2	2	1	1	1	1	2	2	2
60	1	2	1	2	1	2	1	1	2	2	1	2	1	2	1	1	2	1	2	2	2	2	2	2	1
61	1	1	2	1	2	1	2	1	1	2	1	1	1	2	2	2	2	2	2	2	1	2	1	1	2
62	1	2	2	2	1	2	1	1	2	2	1	2	1	1	2	2	2	2	1	2	2	2	2	2	2
63	2	1	2	1	2	2	1	1	2	2	2	1	2	2	2	2	2	2	1	2	2	2	2	2	2

Appendix I

Situational Context Cards

When a customer calls with an unexpected problem and request a team member…

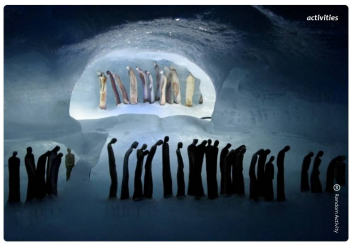
… I should be the one who is contacted.

… I expect someone else handle this call.

In your work setting, do you prefer to observe…

… a colleague.

… yourself.



In work based activities, I would rather prefer a…

… lack of freshness and spontaneity in my day-to-day work schedule.

… lack of a stimulating and energizing project.

In a team based conflict, you prefer to be…

… more firm and sharp to express your thoughts.

… more gentle and conciliatory.

When considering someone who does not fit your team, which sentences would you say…

"… the individual does not have any idea what he/she is talking about, and wasting our team's time."

"… I think this individual does not really fit with our team needs."

What would you say about messy working environments?

… I should have to take care of this mess.

… I really don't care unless it affects my work.

In a team based setting, I am…

… a quick decision maker especially in conflicting situations.

… a late decision maker, I think for a while before I decide.

Consider a meeting with lots of other people from the company, you are waiting for CEO who is late…

… I try to interact with other employees in the room.

… I prefer to think about my work.

In team based decisions, you found yourself against others as a person relies on his/her…

… senses.

… thoughts.

In your team process activities you prefer to…

..deal with what is at hand and get it over with.

..extend the possible options even it's a bit risky.

A teammate of yours came with the idea that something is wrong with a team decision. You need to…

… see the evidence or data to believe something is wrong.

… visualize the big picture and your desires are strong enough to judge.

While evaluating someone's compatibility within your team. You would say…

"….the most important thing is her/his talents, s/he may have to change to fit to our team."

"…s/he is a nice girl/guy, and her/his talents will be improved in the future."

A team based agreement should be in…

… a signed and official form.

… based on a handshake.

I personally enjoy more…

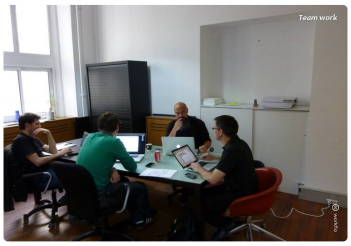
… when we deliver the end product.

… the work in progress stage.

A work client/customer has invited you to a party, do you…

… enjoy meeting with new people.

… prefer to spend my time with the people you know.



When something goes wrong in your team work…

… the facts can save our progress.

… successful communication could save our effort.

In a team, an individual should be able to…

… express himself/herself directly to his/her teammates even it causes personal conflicts.

… first he/she should find a different way to explain the situation to avoid conflicts.

When you are choosing the people you work with…

"… the consistency and stability of their thoughts is the most important factor."

"… harmonious relationships is the most important."

If you need to disappoint a teammate?

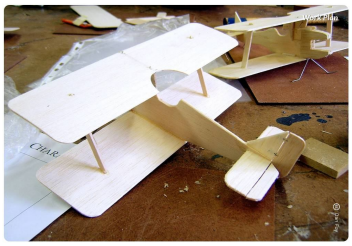
… I will make it directly, frankly and straight forward.

… I will try to be careful not to cause inconvenience or hurt to person.

If I had given a chance to choose, I would suggest my team should use…

"….a plan-driven methodology for development."

"… an agile method will be more suitable."



Hi Guys, I would suggest, we should

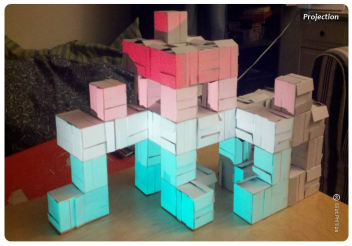
"….prepare a detailed plan before we start spending our precious resources."

"…form a tentative or preliminary plan, which is better, we still don't know many of the parameters."

Your boss ask you for a night out with some of the customers you did not meet before…

… no worries, I would love to, meeting with new people and some chat energize me.

… it is ok, but know that it may reduce my concentration.



We have projected that we are not able to finish the requested module on time…

….we should admit that we fail to perform in accordance with the contract.

… we still have a week, lets wait for a while, and hope that it can be done.

A visionary with an novel idea or a theorist with hypothetical constructs…

…is somewhat annoying.

…is rather interesting.

During a team based discussion…

… evidence suggests that what we learn is mostly from our conflicts.

… I believe, compromise between people for a common ground brings more success.

All software individuals in a team should be …

"…just, concentrate their work, not to the environment."

"… should affected by the sufferings of other members."

A teammate is the one, who…

… must show me my mistakes.

… should try to fit with me.

I would be more relaxed...

... after a software development decision is made.

... before software development decision is made.

While two of your teammates are arguing...

... I enjoy to participate and say what is on my mind.

... although I keep myself out of the conversation, I listen them carefully.

Using common sense in software development is...

... a normal type of behavior.

... not common because some people's common sense is questionable.

A software development team should contribute more to...

... organizational goals.

... be independent and attempt to achieve the goals of the team.

A team leader should be...

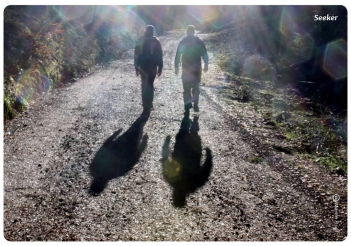
... solid to expose individuals shortcomings.

... tolerant for the members of his team.

A team member should...

"....not easily get excited and show her/his feelings on team conflicts."

"... share her/his feelings with her/his teammates and react to their problems."



I like ...

... solid conclusions or results.

... the possibility of exploring new or unexpected things.

In a conflicting situation, I investigate both sides of the argument...

... in a long an careful consideration.

... in a natural and uninhibited manner.

Considering yourself in a software team, do you prefer...

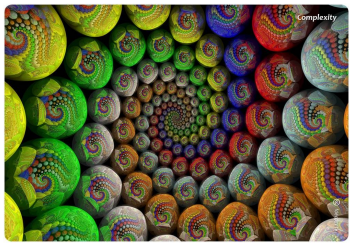
... to collaborate with many peers in a large group settings.

... to have peer to peer interaction.

I am a ...

... practical team mate, I usually find the easiest solution.

... over imaginative person, my solutions sometimes may become unrealistic.



I like...

...the details in a figure, so I can examine it more closely.

...to see a general overview, I prefer to work on details later.

While your boss is assessing your work, which phrase would flatter you most...

... you are a man of logic.

... you are a man of your passions.

You have to write a short sentence to thank all people who contributed to the success of a project ...

"I think the intellectual effort of all individuals bring this work to success... "

"I feel privileged for the opportunity to work with all individuals actively work with us in...."

During the end phase of a task, do you prefer to...

... be sure nothing is missing... .

... move rapidly to the next one

Considering a deliverable date will probably be overdue, do you prefer to...

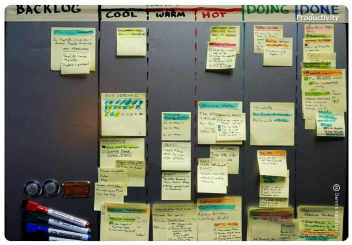
... stay at work until it is finished, to follow up the old schedule.

... negotiate to readapt the schedule

When you are interacting with your team mates...

... I found myself rather talkative.

... I prefer to listen and think about.



When your manager says we need to improve productivity of our team by 1.2, are you inclined to interpret it ...

... in the strict sense of the word.

... as a general objective.

When you are solving a task, do you prefer to work on...

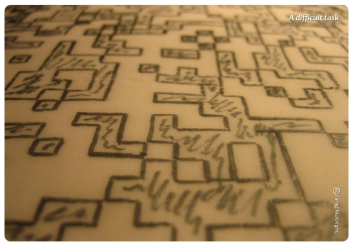
...concrete problems observable immediately.

...potential drawbacks or future improvements.

A reward has to be distributed in the team, you prefer ...

... the distribution should be based on an established performance criteria.

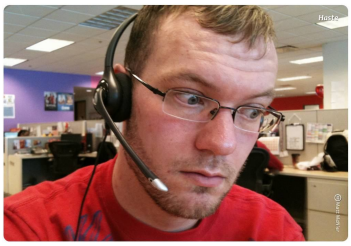
... an equal share among the team mates.



Consider a difficult task that was assigned to an other member of your team, which is overdue ...

... I would not take account of the difficulty to express the problem to the boss.

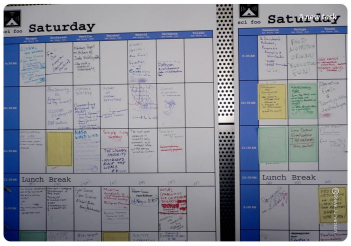
... I would take a part of the responsibility to alleviate his concerns when reporting.



As a team, we need to have new hardware for our work,...

... I intend to do market research before we have our final decision.

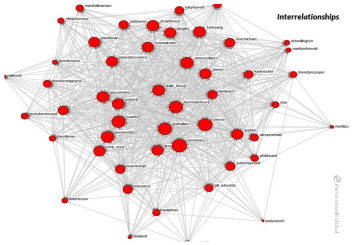
... I inclined to finalize the need and buy them as quick as possible.



Whenever a new task arrived for our team work...

... I am eager to start the job, and to finish it.

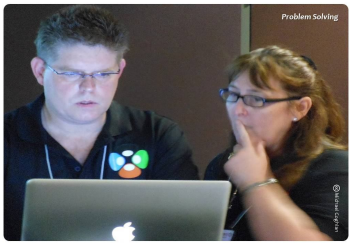
... I will be inclined to examine the new task after we finish the current.



During a work day, do you intend to ...

... interact with your colleagues.

... keep myself focus on my desk.



When you are solving a problem, are you more likely to rely on ...

... your previous hands-on experiences.

... strong and validated principles, which you gain during your work.

The possible way of dealing with a defect in a module would be,...

... examine deeply to come up a solution.

... to find a temporary solution and think about it later.

Consider the situation that two of your team members will be fired, you were asked to vote for...

...I will vote for the one who is less productive without any further consideration.

... I will not vote for people who may have difficulties to find a new job.

Your manager wants you to attend a meeting with her, however your participation may or may not be necessary...

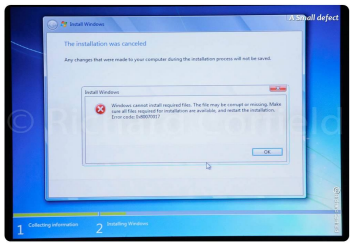
... I will refuse to go because I was already assigned for many tasks which should be finished on deadline.

... I will accompany her because she asked for help.

Considering a project where all requirements have been obtained from stakeholders but might subject to change, and you are planning the implementation…

… I would prefer to have a scheduled and approved plan.

… I would prefer to have a preliminary plan.



There is a small defect in a module, your team has worked hard but cannot find a proper solution…

… I motivate my team to carry on until the issue has been resolved.

… I will suggest to skip for now, and work on other defects that may related to this problem.

Considering yourself participating in a face to face conversation, I will usually prefer…

… to express my ideas.

… to listen others.

When applying a new job, which talent of yourself you prefer to highlight…

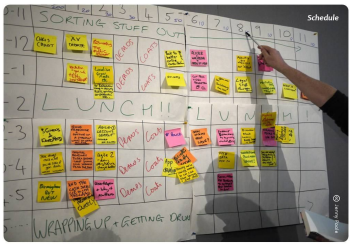
… my pragmatism and sense of reality.

… my creativity and imagination.

Marketing department proposes a couple of slogans for advertising on a product, you will vote for those…

… empathize fundamental features of the product.

… use metaphors to suggest the excellence of the product.



Your team failed to deliver a product on schedule, what would be a bad answer for you…

… I will forgive myself and my team, it is quite common in the software business.

… I will blame myself, and my team will not have any salary bonus for a while.

A market study was set up to determine a new feature for an old product, are you more influenced…

"….with the data gathered by statistical methods."

"… with the beneficial aspects of the feature."

You apply for two different positions in your company, do you feel better when

… coming to the final decision.

… keep your options open.

Your company send you for two week working away from home, do you prefer to…

… have all your work schedule arranged.

… have some space to allow things happen.

As a teammate do you think yourself as...

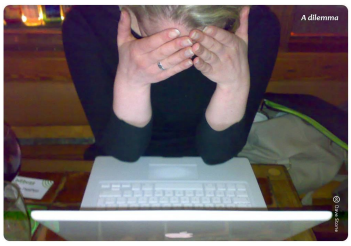
... easy approachable by others.

... standing on your own.

Do you prefer to work on a project ...

... with lot of struggle and technical challenges.

... with lots of opportunities and possibility of a personal education.



You will be promoted for a new position, would you prefer to...

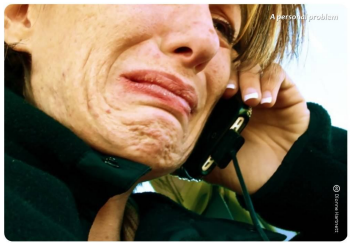
... be a project manager of a team, which is in a moderate competition with other teams.

... be selected as a member of the best team of the company.

What is the best skill for you in a team...

... I will be able to perform my job without being affected by any problem.

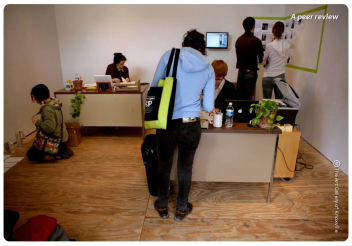
... I will be able to communicate with each team member.



A team mate is suffering a personal problem...

... I don't care because it is not related to work.

... I am also affected by his/her distress.



When performing a peer review do you prefer to...

... highlight and pick up all defects.

... propose an improvement without mentioning the problems.

Do you usually prefer to handle your tasks in...

... well defined ways.

... several different ways.