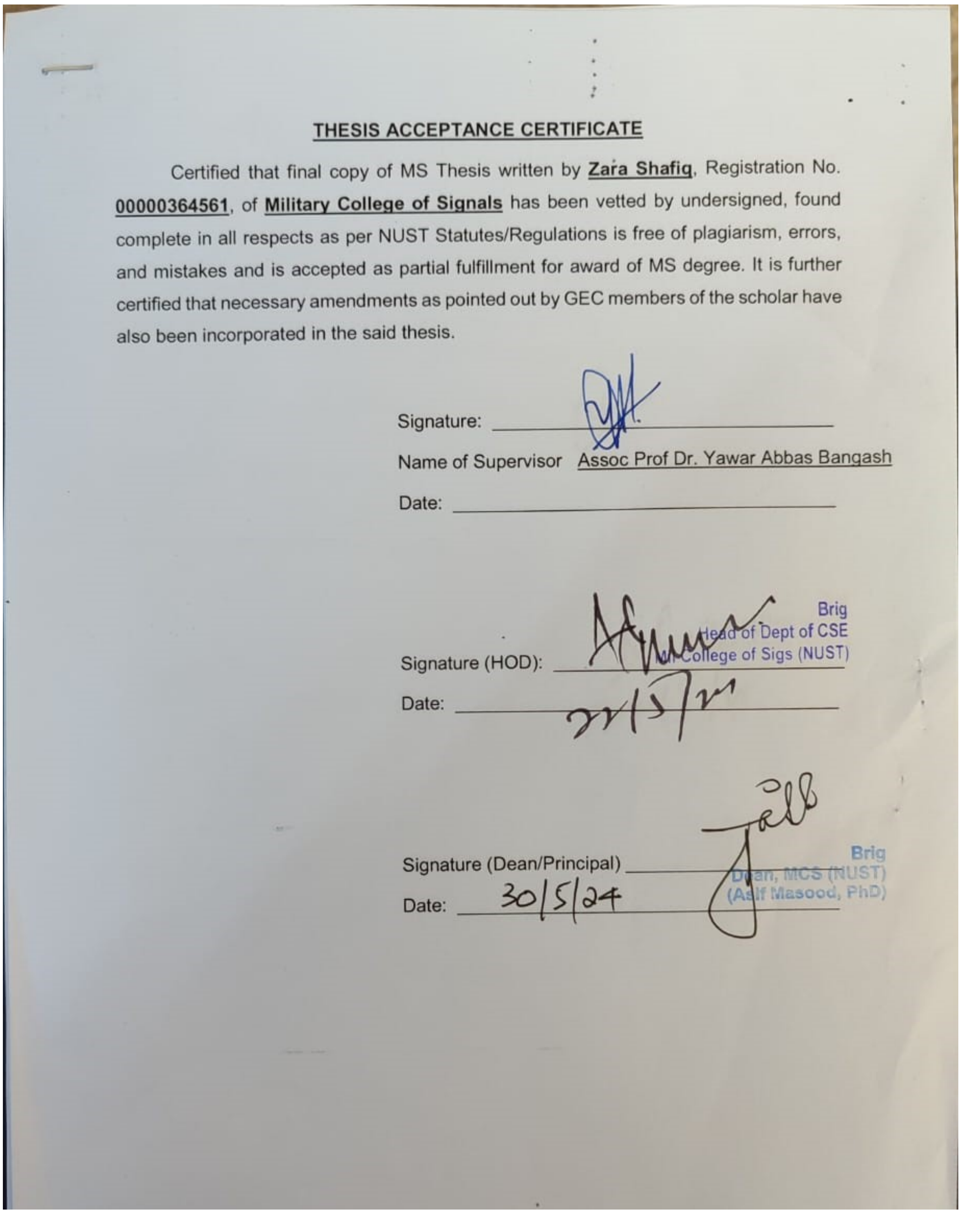
Cost-Benefit Analysis of Software Development Incorporating the Human Factor of Developer

Author
Zara Shafiq
00000364561

Supervisor
Assoc Prof Dr.Yawar Abbas Bangash

A thesis submitted to the Computer Software Engineering Department, Military College of Signals, National University of Sciences and Technology, Islamabad, Pakistan in partial fulfillment of the requirements for the degree of Masters in Computer Software Engineering

(May 2024)



THESIS ACCEPTANCE CERTIFICATE

Certified that final copy of MS Thesis written by **Zara Shafiq**, Registration No. **00000364561**, of **Military College of Signals** has been vetted by undersigned, found complete in all respects as per NUST Statutes/Regulations is free of plagiarism, errors, and mistakes and is accepted as partial fulfillment for award of MS degree. It is further certified that necessary amendments as pointed out by GEC members of the scholar have also been incorporated in the said thesis.

Signature: [illegible]

Name of Supervisor Assoc Prof Dr. Yawar Abbas Bangash

Date:

Signature (HOD): [illegible]

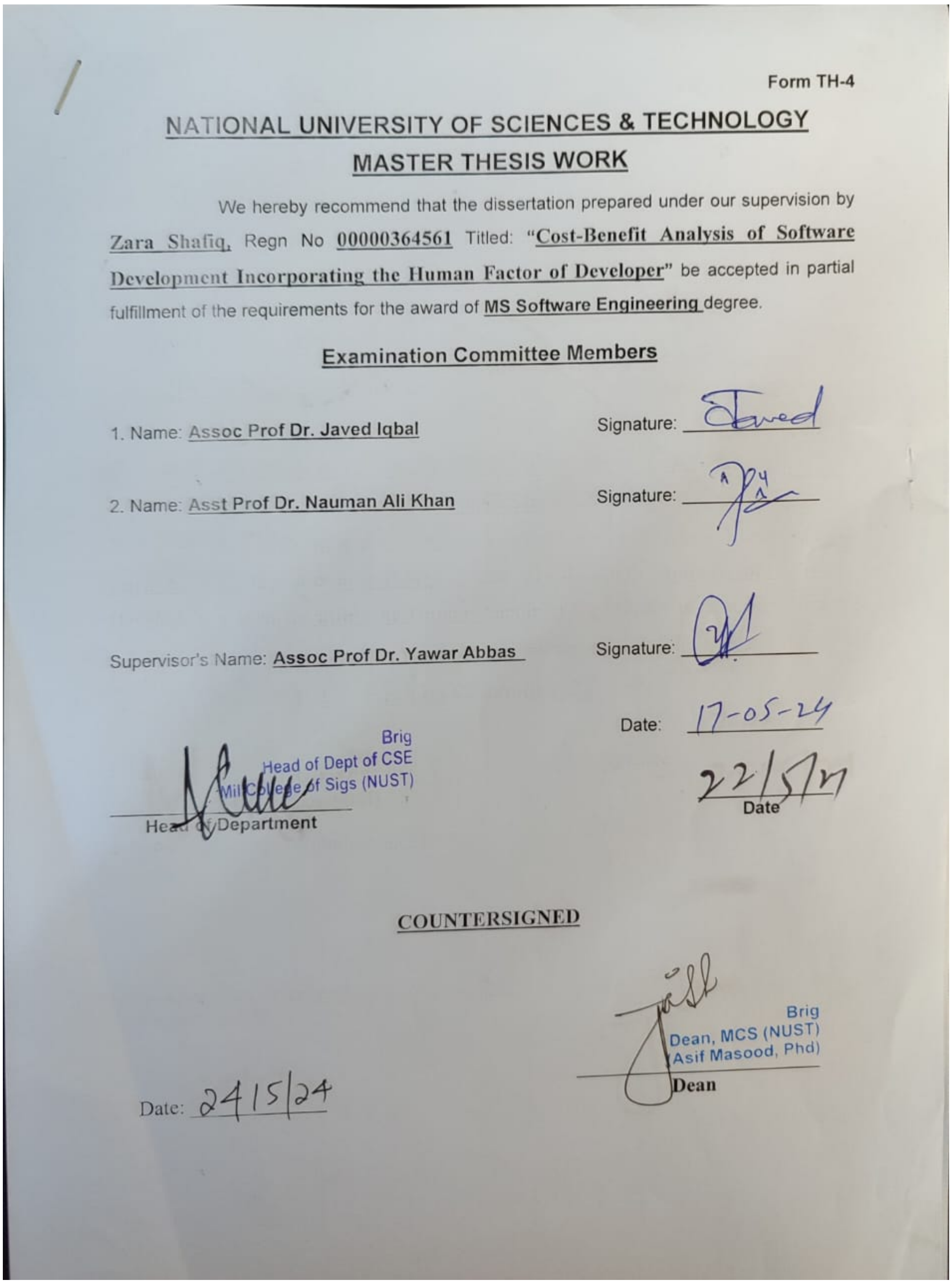
Brig
Head of Dept of CSE
Mil College of Sigs (NUST)

Date: 27/5/24

Signature (Dean/Principal) [illegible]

Brig
Dean, MCS (NUST)
(Asif Masood, PhD)

Date: 30/5/24



Form TH-4

NATIONAL UNIVERSITY OF SCIENCES & TECHNOLOGY

MASTER THESIS WORK

We hereby recommend that the dissertation prepared under our supervision by Zara Shafiq, Regn No 00000364561 Titled: "**Cost-Benefit Analysis of Software Development Incorporating the Human Factor of Developer**" be accepted in partial fulfillment of the requirements for the award of MS Software Engineering degree.

Examination Committee Members

1. Name: Assoc Prof Dr. Javed Iqbal	Signature: [illegible]
2. Name: Asst Prof Dr. Nauman Ali Khan	Signature: [illegible]
Supervisor's Name: Assoc Prof Dr. Yawar Abbas	Signature: [illegible]
	Date: 17-05-24

Brig
Head of Dept of CSE
Mil College of Sigs (NUST)

Head of Department

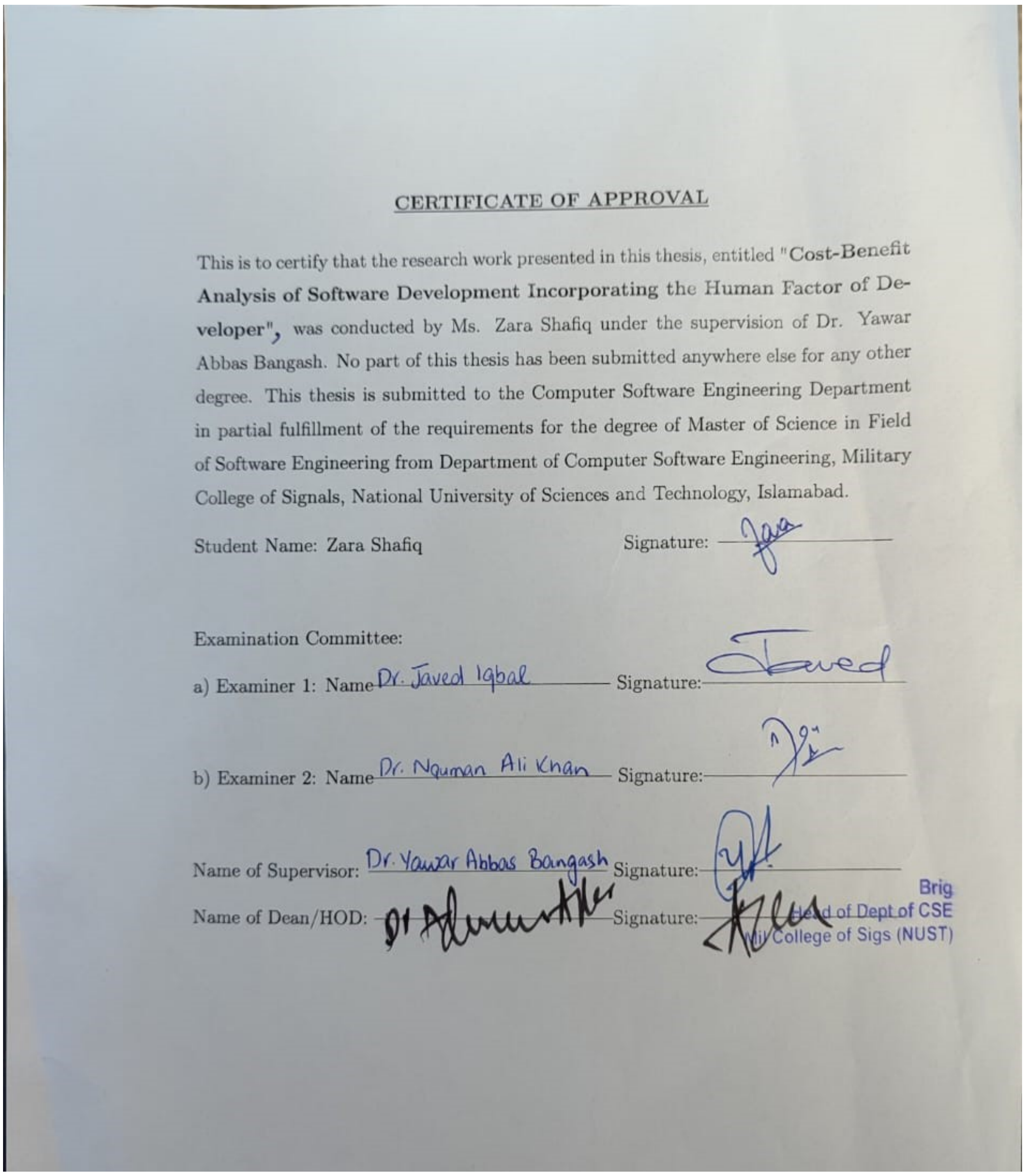
22/5/24
Date

COUNTERSIGNED

Brig
Dean, MCS (NUST)
Asif Masood, Phd)

Dean

Date: 24/5/24



CERTIFICATE OF APPROVAL

This is to certify that the research work presented in this thesis, entitled "**Cost-Benefit Analysis of Software Development Incorporating the Human Factor of Developer**", was conducted by Ms. Zara Shafiq under the supervision of Dr. Yawar Abbas Bangash. No part of this thesis has been submitted anywhere else for any other degree. This thesis is submitted to the Computer Software Engineering Department in partial fulfillment of the requirements for the degree of Master of Science in Field of Software Engineering from Department of Computer Software Engineering, Military College of Signals, National University of Sciences and Technology, Islamabad.

Student Name: Zara Shafiq Signature: [illegible]

Examination Committee:

a) Examiner 1: Name Dr. Javed Iqbal Signature: Javed

b) Examiner 2: Name Dr. Nauman Ali Khan Signature: [illegible]

Name of Supervisor: Dr. Yawar Abbas Bangash Signature: [illegible]

Name of Dean/HOD: [illegible] Signature: [illegible]

Brig
Head of Dept of CSE
Mil College of Sigs (NUST)

PLAGIARISM UNDERTAKING

I solemnly declare that research work presented in the thesis titled "**Cost-Benefit Analysis of Software Development Incorporating the Human Factor of Developer.**" is solely my research work with no significant contribution from any other person. Small contribution/ help wherever taken has been duly acknowledged and that complete thesis has been written by me.

I understand the zero tolerance policy of the HEC and National University of Sciences and Technology (NUST), Islamabad towards plagiarism. Therefore, I as an author of the above titled thesis declare that no portion of my thesis has been plagiarized and any material used as reference is properly referred/cited.

I undertake that if I am found guilty of any formal plagiarism in the above titled thesis even after award of MS degree, the University reserves the rights to withdraw/revoke my MS degree and that HEC and NUST, Islamabad has the right to publish my name on the HEC/University website on which names of students are placed who submitted plagiarized thesis.

Student Signature: [illegible]

Name: Zara Shafiq

Date: 06-06-2024

AUTHOR'S DECLARATION

I Zara Shafiq (Registration No: 00000364561) hereby state that my MS thesis titled **"Cost-Benefit Analysis of Software Development Incorporating the Human Factor of Developer"** is my own work and has not been submitted previously by me for taking any degree from National University of Sciences and Technology, Islamabad or anywhere else in the country/ world.

At any time if my statement is found to be incorrect even after I graduate, the university has the right to withdraw my MS degree.

Student Signature: [illegible]

Name: Zara Shafiq

Date: 06-06-2024

DEDICATION

"In the name of Allah, the most Beneficent, the most Merciful"

This research work is dedicated

to

MY PARENTS, TEACHERS, AND SIBLINGS

for their love, endless support, and encouragement

ACKNOWLEDGEMENTS

I am deeply grateful to Allah Almighty for the strength and passion bestowed upon me to complete this thesis. His mercy and guidance have been indispensable.

My heartfelt thanks to my supervisor, Dr. Yawar Abbas Bangash, for his invaluable advice and unwavering support throughout this journey. His expertise has been crucial in guiding this research to fruition.

I also extend my appreciation to all individuals who participated in my survey and contributed their insights, playing a vital role in the success of this work.

ABSTRACT

This thesis conducts a comprehensive cost-benefit analysis of incorporating human factors in software development. It highlights the critical role of individual characteristics of software developers. The most common human factors impacting the quality of software include adaptability, communication skills, and problem-solving abilities. These human factors play a vital role in enhancing software project outcomes. This thesis employed a mixed-methods approach. Important human factors and cost benefit measures were identified by help of a qualitative analysis of the literature. Quantitative data was collected through questionnaires developed for software developers and managers. The study suggested that human factors significantly boost software quality and team productivity. These human factors require considerable investments which add to the total cost of development. The analysis shows that the advantages substantially outweigh the costs of incorporation of human factors. This research aims to fill an existing gap in literature by providing an analysis on the cost-benefit dynamics of human factors. It also offer valuable insights for optimizing software project management. The research lays groundwork for future exploration in agile methodologies.

Keywords: Cost-Benefit Analysis, Software Development, Human Factors, Software Quality, Team Productivity.

Contents

List of Tables

List of Figures

LIST OF SYMBOLS, ABBREVIATIONS AND ACRONYMS

HF Human Factors
SD Software Development
CBA Cost-Benefit Analysis
OSS Open Source Software
SE Software Engineering
UI User Interface
UX User Experience
QA Quality Assurance
HCI Human-Computer Interaction
API Application Programming Interface
IDE Integrated Development Environment
CI Continuous Integration
CD Continuous Deployment
SQA Software Quality Assurance
SLR Systematic Literature Review
GSD Global Software Development
ICSE International Conference on Software Engineering
CHASE Cooperative and Human Aspects of Software Engineering
IEEE Institute of Electrical and Electronics Engineers
ACM Association for Computing Machinery

Chapter 1

Introduction

In recent years, software has become a central part in our lives and has application in almost every walk of life. Developing software is not only about coding and technicalities, rather it is a deep human process formed by the combination of problem-solving skills and engineering, but we often overlook the human aspect of development i.e. the elements concerning the people behind the programs. The main aim of this research is to explore the role of human factors in the development of software from the lens of cost benefit analysis. In the following sections, after we explore the importance of human factors in software development, we will identify current gaps in literature, and then finally outline the objectives and justification for this study. By attempting to understand the costs and benefits for integrating the human element of development, this dissertation fundamentally aims at contributing to a better understanding of software development process from all perspectives that leads to improving the quality of software developed.

1.1 Understanding Software Development

Software development is intrinsically a sophisticated merge of technology, creativity, and problem-solving. In the modern world, however, software development is pretty much woven into each fiber of modern life: whether these be from business frameworks to educational constructs, personal modes of communication, and of course, entertainment apparatuses. The section will investigate some of the basic methods in software development, with much focus contrasted between the Agile and Waterfall approaches, to lay the foundation in the later discussions of exploring the impact of human factors.

1.1.1 Agile vs. Waterfall Methodologies

1.1.1.1 Agile Methodology

Agile is really a paradigm shift from the conventional, linear approaches to the software development process. Therefore, iterative progress, flexibility, and collaboration with customers are very key to cross-functional teams. Agile is adaptable to change since it is a process that divides the development process into several sprints, in which activities get reviewed and adjusted all the time based on feedback. This approach ensures that the end product closely aligns with customer needs and can rapidly adapt to market changes.

- **Iterative Development:** Short, iterative cycles (sprints) focus on continuous improvement in response to user feedback.
- **Flexibility:**Agile methodologies are highly adaptable, making them suitable for projects with undefined or evolving requirements.
- **Customer Collaboration:** Emphasizes direct communication with clients to ensure the product meets their needs and expectations.
- **Team Dynamics:** Promotes a collaborative and inclusive work environment where every team member contributes to problem-solving and decision-making.

1.1.1.2 Waterfall Methodology

The Waterfall model is a structured, one-at-a-time, and sequential software development approach. The Waterfall model contains non-overlapping phases: requirements analysis, design, implementation, testing, and maintenance. After one stage is complete, it goes on to the next, and it does not go back to the previous stages; hence, much less flexible than Agile. The best-fit project for the Waterfall model is that which has clear and rather stable requirements, with changes very rarely needed or very minimal.

- **Sequential Phases:** Each phase must be completed before moving on to the next, with no overlap between stages.
- **Predictability:** The linear approach facilitates easier project planning and progress tracking.
- **Documentation:** Emphasizes thorough documentation at each stage, providing a detailed blueprint of the entire development process.
- **Stability:** Best suited for projects with clear, stable requirements and a predictable outcome.

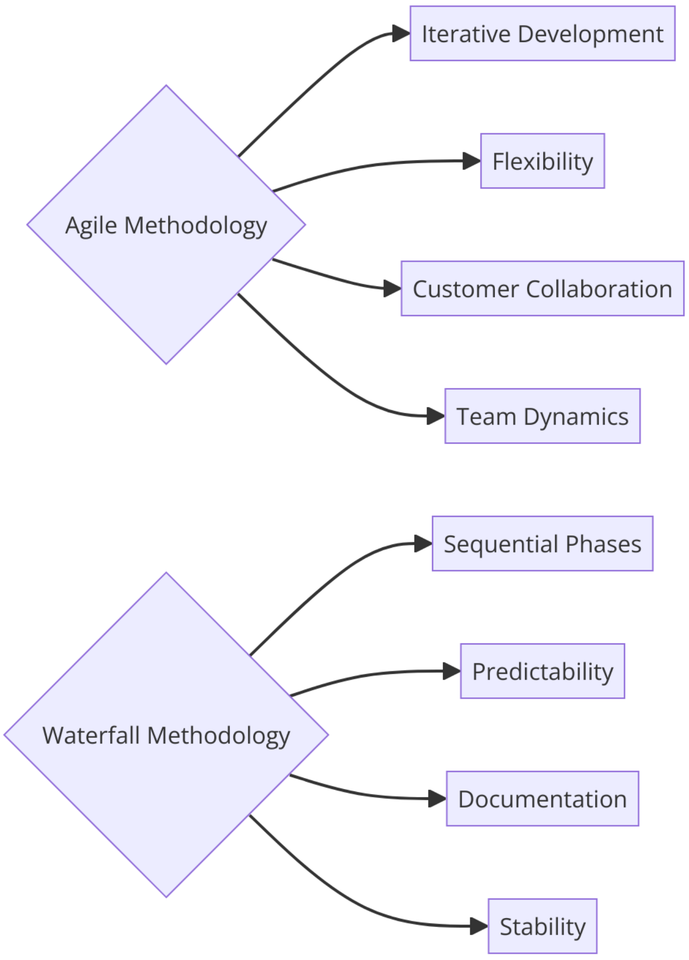


Figure 1.1: A visual comparison of Agile and Waterfall methodologies, illustrating their distinct approaches to software development

1.1.1.3 Comparative Analysis

Where Agile is flexible and opens a more dynamic and collaborative environment in dealing with complex and uncertain settings, Waterfall gives a structured, simple roadmap ideal for projects where the goals are very clear and minimally changed. The choice between Agile and Waterfall really boils down to some very project-specific factors: requirements stability, team size, and customer involvement. This thesis will set a context within which to discuss how integration of human factors can improve the software development process regardless of the approach taken. Understanding these foundational methodologies gives context for the subsequently examined human factors development of software.

1.1.2 Emphasizing Quality Assurance and Testing

1.1.2.1 Quality Assurance (QA)

Quality Assurance (QA) aims at ensuring that the process as a whole, together with its products, complies with predefined norms and requirements. QA practices are preventive in nature; they aim at identifying the defects so that they can be erased. This will encompass a big range of activities, including requirements analysis, adherence to coding standards, and systematic code reviews.

1.1.2.2 Testing

On the other hand, testing is the evaluative counterpart to QA, in the sense that it implies effective software execution, targeted at finding bugs or defects. This is an assurance for the software developed and in operation that it serves its user as intended. Testing can be further elaborated into some types: unit testing, integration testing, system testing, and acceptance testing, where every testing targets some area of the software.

Both QA and testing are significant parts of software development in the lifecycle, supporting the integrity, reliability, and performance of the software product. They reduce risks of failure, enhance user satisfaction levels, and promise the organization that the software will be able to perform under demands reflective of the real world.

1.1.3 Project Management's Critical Function

On the other hand, software development project management extends from planning and implementing toward something more intricate, strategic, and like dancing, coordinating, and adapting toward success in the accomplishment of the software project. This will be accomplished through the project manager, who will be in charge of leading the project from its initiation up to its completion, bearing in mind that software development is usually characterized by complexity and uncertainty.

Actually, project management lies in the capacity to harmonize different teams, manage resources, and, at the same time, set the trajectory for the project in line with the goals that had been intended. Effective communication, risk management, and making sure that all the activities have the involvement of the stakeholders are some of the roles that ensure that a project does not derail but outputs a product that either meets the expectation of the stakeholder or exceeds it.

Project management provides focus toward the delivery of value and collaboration that, in great definition, defines the environment in which software development will take place. It is, therefore, an important element in significant influences of quality and timeliness, and consequently, the success of the produced software.

1.2 Exploring Human Factors

Human factors are all the ways through which human characteristics influence or are influenced by participation in the software engineering process. These will constitute a number of considerations, from cognitive and physical capabilities of the employees to the social dynamics and the organizational culture at the workplace.

1.2.1 Defining Human Factors

Human Factors, sometimes exchanged with Ergonomics, refer to the understanding of how man interacts with several systems, and using this understanding for optimization in both well-being and system performance. Development will thus imply the creation of environments and tools that will enable practitioners to perform development activities effectively and in a more productive manner. The following shall define basic principles of human factors that will set a stage to delve into just how these principles have impacted software development. The importance of the human factor continues to play a paramount role in the wide expanse of creativity, innovation, and collaborative problem-solving. Individual differences, culture of freedom of speech, and development of resilient teams are some of the factors where the human factor is very essential. It is, however, the confluence of these that often dictates efficiency, effectiveness, and overall health in the software development process.

1.2.2 Cognitive Abilities and Individual Traits

This is the human factor that recognizes the essential fact: cognitive abilities are manifold and individualistic. Cognitive abilities like attention, memory, and problem-solving skills bear on developers' way of approaching the solution to the task in software development. The major traits that affect collaboration and work preferences, on the other hand, have got to do with the personality types and learning styles. Inclusive software development will certainly enable the divergent abilities and traits to add or create more value while enabling them into innovative solutions.

1.2.3 Impact of Psychological Traits on Decision-Making

Psychological traits and attributes—for example, risk tolerance, motivation, and resilience to stress—make a very important role in decision-making within software development. Furthermore, the following characteristics will lead to how the individual member will make decisions of the teams. This section presents a discussion of how psychological influences impact the outcomes of software development and the importance of leading toward a developmental environment that minimizes negative psychological well-being.

1.2.4 Communication, Teamwork, and Collaboration

Effective communication, teamwork, and collaboration are prerequisites for success in any given software project. In human factors, focus is given to the foundation of developing an environment of open communication, the strength of diversity of the team members, and effective collaboration. The core mission of this paper lies in emphasizing, through this part of the introduction, the indispensable role of such social dynamics in reaching project objectives, mitigating risks, and ensuring the delivery of first-class software products.

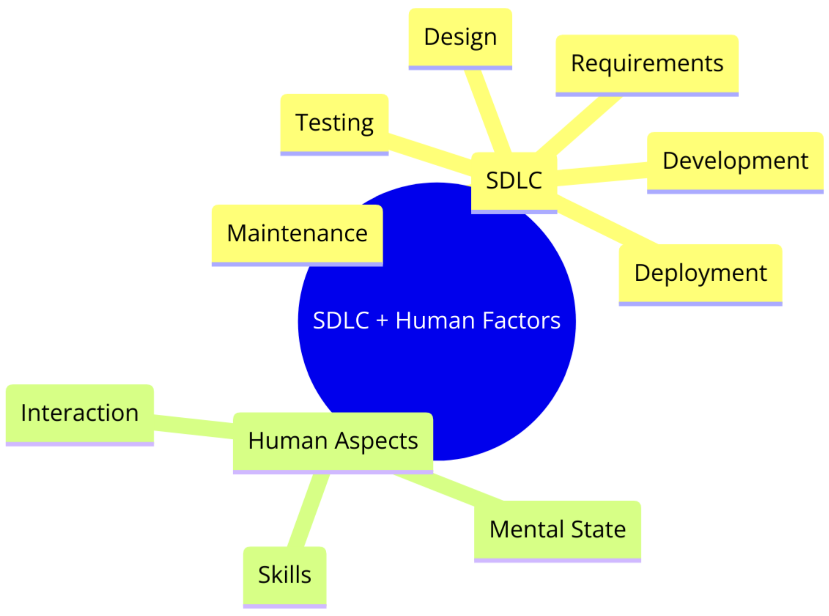


Figure 1.2: The Costs and Benefits of Incorporating Human Factors in Software Development

1.3 Advantages of Integrating Human Factors

A software design that would include human factors within its development signals a complete shift towards more empathetic and user-focused methodologies. This strategic inclusion really brings it from just technical excellence to effectively bridging human needs with technological solutions. The benefits, on the other hand, that result from such integration are deep; they range from improved team dynamics and higher satisfaction of the stakejson to improved software quality, streamlined development processes, and improved job satisfaction and retention. In so doing, it is able to coordinate with intrinsic human elements of software but adds substantially to efficiency, effectiveness, and success in developing a project. Analyzing each of these more in-depth gains, it will be widely manifest that the human factor is not something added just to the scope as value but as one of the bases for supporting and stimulation of innovation and excellence in the field of software development.

1.3.1 Enhancing Team Dynamics

Therefore, human factors in software development are an integral component of adding substantial value to the dynamics of the team. Work with projects that identify and account for the skill diversity, requirements, and communicative style differences among its members often results in improved work atmospheres to be more cohesive and collaborative. This section explains how such an inclusive approach derives better problem-solving abilities, innovation, and eventually translates into better project efficiency - emphasizing the very important role of human factors in team building into high-performing teams.

1.3.2 Boosting Stakeholder Satisfaction

Some of these comprise the human factors of the development process, which either favors or benefits the respective team in question or improves satisfaction for stakeholders. This section of the paper will outline in detail how understanding and meeting the users' and stakeholders' needs through a human-centered design approach give rise to end software products that better serve their intended purposes, therefore going ahead to increase acceptance and use levels of the same among the users and stakeholders.

1.3.3 Elevating Software Quality

The greatest benefit in embedding the human factor means the improvement of the quality of the software. Practices that would encourage the spirit of user-centered design, continuous feedback, and ergonomic considerations in the process of development would ensure software above its functionality—their intuitiveness, reliability, and friendliness. This section will explore the relationship human factors have with software quality, with a specific reference to the direct effect on usability, accessibility, and general user experience.

1.3.4 Streamlining Development and Reducing Costs

Integrating human factors into software development processes provides an approach in which human resources will be allowed to facilitate organizational productivity, probably with potential costs being saved. Such a design approach may even allow human beings to reduce the risk of expensive errors through optimized workflows and communication, less rework, and development cycles getting more efficient. This part will elaborate on how taking into account the human factor will enable one to ensure more efficient resource use and, therefore, reduce costs in the long run.

1.3.5 Promoting Job Satisfaction and Retention

Thirdly, the inclusion of the human factor in software development contributes a lot to the job satisfaction and retention rates of developers. Realization and addressing of issues related to the work environment, work-life balance, recognition, and growth opportunities may, in turn, further aid in having more motivated and committed workforces. This section will explain how improvement of organizational culture towards a positive one that values the human factor might lead to reducing turnover, increasing team engagement, and finally, productivity.

This opens up the advantage of including human factors integration and brings out the overall advantage in the different areas of software development. From improved dynamics of the team to improved software quality and improved job satisfaction, human factors integration emerges as an important strategy towards superior outcomes of software development.

1.4 Challenges in Human Factors Integration

As much as the process will be beneficial in seamlessly integrating human aspects into software development, it is filled with a lot of challenges. Many of the challenges confronting it are drawn from the fact that human behaviors are complex, and software development teams have delicate dynamics. These challenges must be addressed by organizations that want to reap the full benefits from human-centric approaches. Diversity of team dynamics creates complexities, while complexities on effective communication strategies and the challenges posted by aligning project management methodologies with human factors rank principal challenges in the integration process. That is particularly the balance sought in the dealing with challenges of innovation versus user-friendliness, covering an all-inclusive design where software development is concerned and training to cater to changing needs for the purposes of development, given continually increasing and changing technologies. All these have to be overcome strategically with full focus on empathy, adaptability, and constant learning within development.

1.4.1 Necessity for Enhanced Training

Essentially, this is the main reason human factors integrate; it has to cost a huge investment in training of staff, both in finances and time. Essentially, the process of effective training is expensive and very important, meaning it is important in instilling the new process to them and should consider cost versus benefit.

1.4.2 Managing Diverse Teams

While diversity in the software development teams may be considered good for fostering innovation, it adds in place complexities for the communication and cohesion of the teams. Management strategies that foster multiplicity in opinions and at the same time work for the good are pertinent for the success of an organization.

1.4.3 Balancing Productivity with Well-being

The inclusion of human factors is made with the hope of increasing productivity and job satisfaction, but again, it increased the danger of stress and burnout. Striking a balance that will allow very high productivity yet ensures the well-being of software developers is a tuned fine problem.

1.4.4 Cognitive Biases and Decision-Making Errors

Cognitive biases and judgment errors could create havoc during the software development process and lead to outcomes that are far less than optimal. Identifying the biases and possibly reducing the impact is very important toward making any informed decisions and being efficient.

However, the urge to integrate human factors in the process of software development is motivated by the immense benefits it yields. Strategic planning, which should include continuous training about good communication and dynamics of the team, is one of the approaches to solving such challenges.

1.5 Research Gap: The Need for Cost-Benefit Analysis

Although the human factor has long been recognized as one of the key issues in software development, to date, a critical gap in the current literature exists precisely toward a comprehensive cost-benefit analysis from its integration. This gap, therefore, gives an important place to the very pressing need of detailed research that will quantify not only the financial but also the qualitative impact of integrating human factors in software development processes.

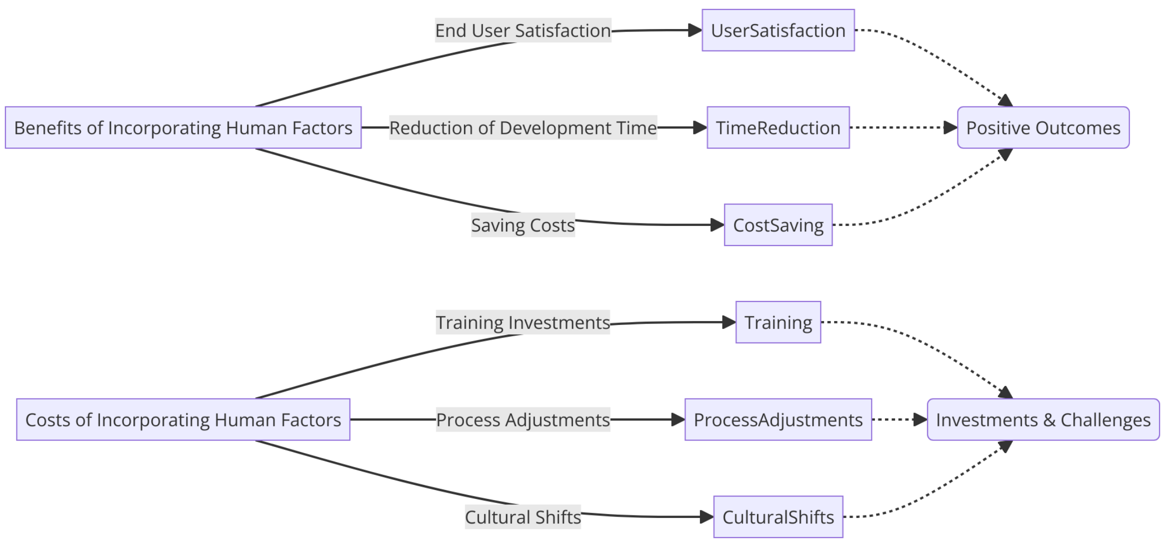


Figure 1.3: The Costs and Benefits of Incorporating Human Factors in Software Development

1.5.1 Uncharted Financial Implications

A critical point of financial view to the integration of human factors, from the initial investment in training and development to long-term implications of the productivity and success of the project, is thus rather insufficiently explored. Most of the literature available focuses much on the qualitative benefits of how it enhances team dynamics and quality of software but goes the extra mile in details for a financial analysis. This is an outstanding omission since it leaves a very huge gap in understanding what the true economic value of human factors is in software development.

1.5.2 Qualitative Benefits Versus Quantitative Costs

While the reported benefits of integrating human factors are manifold—from improved team efficiency to improved software quality—overwhelmingly, most of these are reported and discussed in qualitative terms. Without a quantitative analysis, the organization is hardly in a position to make sense of whether or not the investment of its resources in integrating human factors into the design is worthwhile.

1.5.3 Addressing the Gap Through Empirical Research

It is just this gap that this research seeks to fill by conducting detailed empirical analysis to the costs and benefits associated with human factors integration in the software development process. It tries to take a detailed view so that it will be beneficial for stakeholders in making decisions. This will be through giving an understanding of the tangible and intangible facts of the integration.

1.5.4 Potential for Transformative Insights

This is going to fill in the research gap and possibly change how organizations view and apply human factors in software development. These cost-benefit dynamics might provide a clearer picture guiding more strategic, informed, and effective integration of human aspects into the development process.

In such context, it should be mentioned that the need for conducting an extensive cost-benefit analysis is proving to be a noteworthy research gap in the study being made, related to human factors in software development. Such a gap, if bridged, may unfold newer insights within not only the economic but also the qualitative effects of human factors integration, and this may only help in software development practices becoming more effective and efficient.

1.6 Objectives and Justification of the Research

Basically, the main purpose that this proposed research would seek to achieve is that of following systematic cost-benefit analysis towards discussing the role and impact of human factors in software development.Specifically, this research aims to:

1. **Identify Key Human Factors in Software Development:** Most importantly, the results of an in-depth literature review are going to be presented together with empirical data from software developers and managers. This is to establish which human factors are referred to most often and their importance in the software development process. In that line, apparently, the objective shall be to ground the design and dissemination of the questionnaire on such aspects as adaptability, communication skills, and problem-solving abilities.

2. **Analyze the Costs and Benefits of Incorporating Human Factors:** Analyze the economic implications quantitatively of integrating the identified human factors directly and indirectly into software development practices. This would measure observable outputs, all of which are based on responses to questions, in the form of defect rates, project delivery schedules, or, more generally, software quality.

3. **Evaluate the Balance Between Costs and Benefits:** It will also help establish whether, under human factor integration within software development

practices, it can be assured that there is an overall positive net effect, with emphasis on a point of balance between the associated costs and derived benefits. This objective is going to be realized through statistical analysis of survey data to correlate investment in human factors with improved software quality and other outcomes.

4. **Propose Recommendations for Effective Integration of Human Factors:** Base the recommendations on effective strategies that software development teams and their managers can implement for human factors with clear methods of achieving benefits while minimizing costs. The recommendations will therefore be based on both literature review and empirical data analysis.

1.6.1 Justification

Key of this kind of research remains in providing an all-around view with regard to the understanding of the many impacts and causes of human factors in software development. Many of these works pointed to the importance of human elements in the development process, but systematically analyzing the same by using cost-benefit analysis is mostly unexplored. This research seeks, through bridging this gap, to bring about a more developed understanding of how human factors contribute to not only the qualitative sides of software development, such as improved team dynamics and quality software but also to provide quantitative measures that include cost savings and efficiency gains.

Furthermore, such work shall take into consideration practical challenges that software development teams are likely to face when trying to integrate human factors into their workflow. In this regard, it will assist practitioners in making evidence-informed decisions for the realization of the economic implications needed in both approaches to be integrated into their practice, thus improving the efficiency of the developed software product and raising the quality level.

Research Questions

1. What are the key human factors in software development as identified by literature and industry practices?

2. What are the specific costs associated with incorporating these human factors into the software development process?

3. What tangible and intangible benefits can be realized from the integration of human factors in software development?

4. Do the benefits of incorporating human factors in software development justify the associated costs, and under what conditions?

1.7 Significance of the study

This study shall be of importance to certain stakeholders of the software development domain, such as academic researchers, software development teams, project managers, and organizational leaders. Specific to cost-benefit dynamics of integrating human factors within software development processes, the contributions of this research will be:

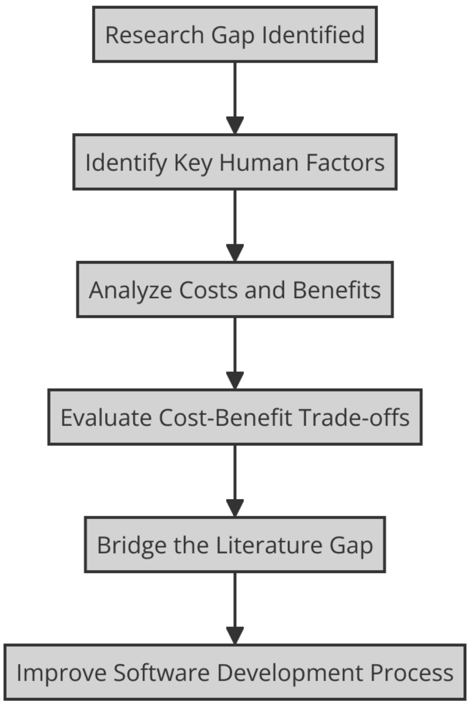

Research Gap Identified
Identify Key Human Factors
Analyze Costs and Benefits
Evaluate Cost-Benefit Trade-offs
Bridge the Literature Gap
Improve Software Development Process

Figure 1.4: Abstract Diagram of Research Workflow

1. **Academic Contribution:** This study is of critical importance in filling one of the major gaps in the current body of literature bearing in mind that it follows the manner in which human issues in software development are included through economic implications both quantitative and qualitative in nature. Further, the study utilized the larger cost-benefit analysis framework in adding to the knowledge base in existence in the area of software engineering and human factor research. It was also provided from empirical evidence that theories that

postulate the integration of human aspects with system issues can build software quality and development efficiency.

2. **Practical Implications for Software Development Teams:** The research provides relevant practitioner insights that could aid the management of the human factors in a strategic response towards realizing better outcomes in software development. The study will help teams focus on the most critical human factors that are most influential and predictive of their economic consequences so that success, in guiding the team to prioritize the areas for both improvement and investment, will be achieved. The results obtained may tentatively guide software development teams on the practices adopted and utilized to provide a positive working environment and the economic need thereof.

3. **Decision-making Aid for Managers and Leaders:** It further provides the managers and organizational leaders with essential information that plays a strategic role in taking investment decisions in training dollars, teaming and project management approaches that support cost-efficient resource allocations in maximizing the return on investment.

4. **Policy and Framework Development:** More importantly, the reed study can be helpful in informing the policy level of individual organisations and the industry as a whole, by uncovering the salient role which human factors can have on impacting continued improvement in software development. It helps in the development of the framework that the human factors which together with traditional software methodologies support the practices that uplift human well-being as well as system performance.

5. **Future Research Direction:** Finally is this study pointing to areas where less data is and proposing new methodologies with an aim for them to be done and raising a question for them to be possible. It, thus, paves the way for the multidisciplinary exploration of the area of the insertion of the scratch pad to the interface area of human factors and software development, hence inquiry on the interaction of those elements in different sorts of contexts and types of projects.

The importance placed on this study further looks at the role of human factors in the development of software. The paper attempts to contribute towards understanding how human-centered approaches can prudently be integrated pragmatically into software development practice, finally contributing towards betterment in qualitative software products, relationships of the team, and project success.

1.8 Thesis Outline

This thesis is divided into 7 chapters.

- Chapter 1: This chapter includes the basic introduction, establish the objectives and primary contribution of my research work.
- Chapter 2: This chapter describes the study of existing research on incorporating human factors into software development.
- Chapter 3: This chapter highlights the research methodology of my thesis.
- Chapter 4: This chapter highlights the data analysis of my research.
- Chapter 5: This chapter studies the analysis of the data collected from my research.
- Chapter 6: This chapter presents the conclusion and findings of my research.

Chapter 2

LITERATURE REVIEW

2.1 Introduction

With the recent advances in the development of practices related to software engineering and the correlatively growing importance of human factor in the development of software engineering, understanding the deep connection between human factor and software development is very important. This literature review aims at providing a comprehensive overview of the relevant literature for our research, with a particular focus on the human factor and factors that might translate as the potential costs and benefits of the literature.

2.2 Overview of Software Development

Modern software development has wholly become a part and parcel of each industrial sector and commands a considerable portion in the present world. These tasks are used in a series of the most comprehensive in order to build a software product, ranging from requirement gathering, designing, coding, testing, debugging, and maintaining the system [3]. It has completed the software application meeting the defined business or personal objective.

Leading paradigms, with respect to this to a greater extent, are approached by waterfall and agile methodologies. The software development methodologies change historically over time in compliance with different methodologies and models accepted in response to the predominant need and context of technology [9]. Most cases tend to reveal that Waterfall and Agile methodologies are some dominant paradigms in this respect. Waterfall model is the first approach proposed in the system development process, where the sequential flow of steps was from requirement analysis to system design, implementation, testing, deployment, and maintenance, suitable for static definite projects. The following are key tasks involved in the software development process:

- Requirement Gathering
- Designing
- Coding
- Testing
- Debugging
- Maintaining

This comes in sharp contrast to the Agile methodology preached by modern software development, which propagates flexibility and the constant interaction of the development team along with the counterpart stakeholders. The approach supports the increment development of products or services with a lot of focus based on the collaboration of the customer and their ability to respond to the respective changes. Among the most popular methodologies applied under Agile are Scrum and Extreme Programming (XP). Coming with unique strength, these have contributed to the shaping of how software development is done.

The quality in software is very key to the concept of software development. Quality is the degree of specification which the requirements for a system, component, or process [6]. The quality assurance practices toil under the software development course to ensure the deliverable product is of high quality. These involve systematic planning and the set of activities are implemented so that assurance can be given to the customer that the specified quality requirements will be met [7]. Testing, as we have noted, is a key activity in quality assurance needed for the validation and verification of the software product regarding the implementation of similar characteristics with the required functionality [8].

Moreover, the role of project management in software development can never be underplayed. Project management within this context is the knowledge, skills, tools, and techniques applied to meet the project requirements [5]. The project manager plans, executes, and administers the project to completion within schedule and budget. In essence, project management tries to ensure that the delivered software product is of high quality and within the scope, budget, and time [10].

2.2.1 Software development methodologies (e.g., Agile, Waterfall)

Since the inception of the software development industry, several methodologies had been observed, which gave birth to back the software development procedure. Agile and Waterfall are considered two leading software development approaches that are fundamental in project management [6].

Table 2.1. Comparison between Agile and Waterfall Methodologies

Criteria	**Agile Methodology**	**Waterfall Methodology**
Approach to Development	Iterative and incremental	Linear and sequential
Requirements	Frequently changing or not well-defined initially	Well-defined and stable
Project Phases	Divided into short cycles called sprints	Sequential phases (e.g., requirement, design, coding)
Flexibility	Emphasizes adaptability and response to change	Limited flexibility
Customer Collaboration	High emphasis on customer feedback and involvement	Limited customer involvement until the final stage
Team Communication	Frequent communication and collaboration	Less emphasis on constant communication
Progress Monitoring	Regular check-ins and adjustments during each sprint	Milestones and progress measured at the end of each phase
Documentation	Less emphasis on extensive documentation upfront	Detailed documentation at each phase
Risk Management	Risks are identified and addressed iteratively	Risks are addressed in the early planning stage
Time and Cost Estimation	Iterative estimation and adjustment	Detailed estimation upfront
Quality Assurance	Continuous testing and quality assurance activities	Testing activities at the end of each phase
Suitable Projects	Projects with changing requirements or high uncertainty	Projects with well-defined and stable requirements

Agile methodology is a software development process, with an iterative development approach that bases its foundation on collaboration, customer feedback, and small, rapid releases. It is the subdivision of the project into smaller tasks called sprints. Sprint is a one- to four-week cycle during which a team works on a set of tasks that the team has committed to complete beforehand. This Agile process—with more flexibility and possibilities for adaptation to changes—is most helpful in the current ever-quickening technology environment. It should be highly recommended for the cases where requirements are supposed to be changing on a high-frequency basis or are ill-defined at the project start [3].

On the other hand, it is a kind of software development life cycle that is linearly sequential, whereby one has to ensure that each and every phase of the project is completed before he or she moves the process to the new one [4]. This methodology suits best for clear requirements and for supposed minimal changes during development to be realized. Waterfall approach evidently structures and hence easily monitors and controls the project progress [4].

This application has its strong points and weak points against both methodologies: Agile finds itself most appropriate for high adaptability and high communication-need projects, while Waterfall is most useful for unchanging requirements and predictable results projects [8]. The right methodology, in this case, will be the factor of the needs, goals of the project, preference by the development team, and experience. Project management also calls for knowledge in some form of software development methodologies like Agile and Waterfall. The big difference is that in methodology, the project runs, making a big difference between the success of a project and the quality of the final product.

2.3 Key concepts and practices

Concepts and practices in the development of software are very critical. They guide the process to become successful. Many but here we focus on three crucial ones: modularity, documentation, and testing. These are very crucial to control and manage the software development process [5].

Modularity is one of the more refined ideas that draw a distinction between software made up of separate independent modules, each with a specific task. This is one of the many benefits that come with it. It is easily debuggable and, at the same time, maintainable and testable. Following the modular structure and designed interfaces, multiple developers can now work on the modules in parallel, which further increases the pace of development [6][11][15]. **Documentation** is the most important software development practice as it is the written record for all the software. It covers almost all details, starting from design, code, and use cases. Proper documentation makes another developer understand the software well and can also help in the future enhancement of the software. Documentation is like a manual guide for software, which is very necessary for maintaining the software [11].

Testing is done to find any bug/error in the software before it is released. It is very important because it ensures that software is working as expected. It can be done in any way like unit, integration, system etc. It helps increasing the quality of software and reduces the risk of failing in real-time usage [13].

The key concepts and practices form the backbone of any software development process. They guide the development process and ensure that the software is of high quality and meets the user requirements.

2.3.1 Importance of quality assurance and testing

The role played by quality assurance and testing in software development cannot be overemphasized. Reliability and functionality in the software developed [13]. Quality assurance and testing are the most important activities in the software development life cycle, helping in the identification of software defects and removal of the 'mistakes' from the software product [15]. This holds prime importance in increasing the overall quality and performance of the software.

Quality assurance is one of the processes and procedures meant to ensure that the software is developed following the predefined standards and requirements. It focuses greatly on defect prevention and error during the development process. It includes quality assurance activities like requirements analysis, code review, and adherence to coding standards [11]. Therefore, with the use of such practices, the software development team is able to catch and correct the happening errors or issues in the occurrence initiation stage in the software development cycle, thus saving lots of time and efforts in the long run.

Testing is the component of the central procedure in determining and validating the functionality and performance of software. It involves the execution of many test cases and test scenarios for purposes of confirming defects and ensuring that the specified requirements for the software are met. Requiring, among others, a strong preference for practical exercises, involvement, interaction, skill development, and reality-based situations [14]. Testing can be applied at varied levels, such as

- unit testing

- system testing
- integration testing
- acceptance testing

. Every layer of testing assists in defining certain sorts of defects and seeing to it that the software can function according to the objective.

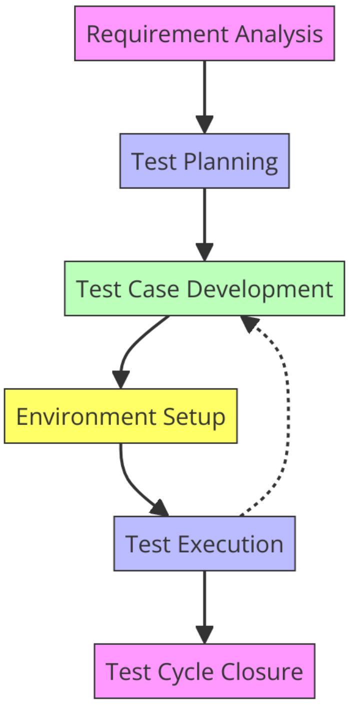


Figure 2.1: Flowchart of the Software Testing Lifecycle, illustrating the systematic process of ensuring software quality through testing

A number of advantages of proper quality assurance and testing practices can be brought into software development projects.

1. First and foremost, it contributes to raising the **general reliability and stability level** of the software by decreasing the probability of software failures and malfunctions [13]. This is very important in mission-critical applications, whereby slight defects can cause great losses.
2. Second, due to quality assurance and testing, it **solves compatibility problems**, enabling the software to run smoothly on various platforms and environments [15].

3. This is a process of testing that eventually leads to overall **customer satisfaction** through the delivery of a software product that meets or exceeds their expectations [11]. It is a quite vigorous testing process and allows the software developers to spot and fix potential usage problems, hence considerably improving the overall user experience.

4. Quality assurance also helps in testing and contributes to a **reduction in cost** because it traces the defects and fixes them during the development. It is more expensive to make changes during the development stage rather than after the deployment of software [16].

But among so many benefits that it can offer to the software development process, quality assurance and testing process come with their unique challenges. "Comprehensive testing can be hard to conduct due to

- limited resources
- time pressures
- a dynamically changing set of requirements
- selection and identification of test cases for efficiency and effective testing

This, therefore, brings into context the importance of having quality assurance and testing as part of software development. All the quality assurance and testing efforts are geared towards improving the general reliability and functionality, hence the customer satisfaction with the software product. Though they bring some challenges with themselves, their benefits for enhanced quality, reduced failures, and cost saving do justify their inclusion in the software development process.

2.3.2 The role of project management

Project management is very important in software development. Project management is a field, which is dealing with smart planning and organization, and effective control of different resources applied in an appropriate way to reach the defined objectives [17].

In general, for software development projects, given that they are

- complex by nature
- involve interrelations
- have high levels of constitutive acceptance of changes

Because of these reasons ,the circumstances regarding them are such that project management of the constituent is very important. For instance, a system that manages the flow of development, balances team working, and ensures deliverables of work reach customers at their wanted quality levels on time and within the right budget, as planned [23].

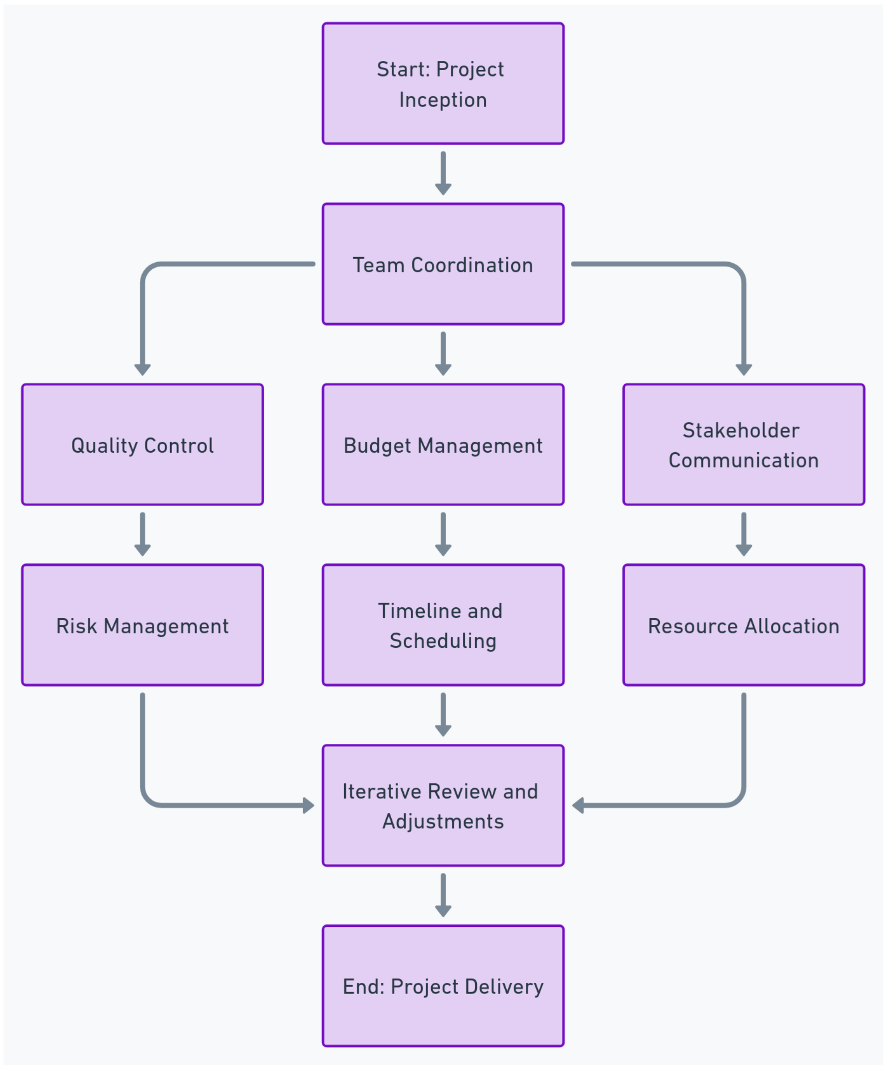


Figure 2.2: Comprehensive representation of the various functions and responsibilities of project management within the software development lifecycle

A software project manager heads the development in the software. He ensures the developed software has been done in regard to the requirements as laid down by the client or stakeholder. It is like traveling, where analysis, planning, and close watching of the progress of the project assure that every phase of development is carried out flawlessly and on time [22].

Project management is actually not just about leadership; it entails managing the whole team that contributes, harmonizing efforts, and solving conflicts among contributors within the team. Problems and issues that accompany the development process are intrinsic, and the project manager is the one to lead in risk and issue management [25].

They help a great deal in also communicating the project plan; this by all means ensures that all team members are informed and kept intact with projects progress and if in any case there is an alteration, then the project plan is altered and communicated in an efficient and fast manner. This paper offers a channel of communications between the development team and the customer, coming out effective and smooth communication process [18].

In summary, the need of project management in software development work cannot be overemphasized. It is instrumental in ensuring that such projects maintain and finish timely, budget constrained and to the satisfaction of the client. And most of all, plays a vital role in promoting harmonious working relationships and by extension proper communication between or among team members as well as the stakeholders. That said, project management is an intricate undertaking and demanding that it requires a great deal of skills and expertise.

2.4 An introduction to Human Factors

2.4.1 Definition and explanation of human factors

Human factors often, interchangeably, refer to **ergonomics**, the profession, usually, refers to

Definition 1. *Understanding the interactions between humans and other elements within a system, and the application of theory, principles, data, and methods to design, in a way to optimize human well-being and overall system performance [26].*

Indeed, "human factors" is a term principally used in the language of industrial design and "ergonomics" has been the more popular term within the human factors community in work environments. However, the goal is the same regardless: create systems, processes, and tools that fit the users and operators rather than crating the force-fit with unproductive humans who simply cannot comply with the obvious errors in the design of a system.

2.4.1.1 Key Elements in Human Factors

1. **Cognitive Abilities and Load:** The conduct also includes the understanding and accounting for the physical and cognitive abilities and constraints and inclinations of developers of software within this context of software development. Inclusive are considerations of motivation, satisfaction, and testing effects of stress and fatigue [24]; this boils down to very important variables likely to have influence on the effectiveness of a software development team. Human factors research sets out to uncover and comprehend human characteristics relevant to the design of a system. An example of how knowledge of a cognitive load may be used is that it could be used in the design of a more instinctive, less error-prone software system. On the contrary, such guidance often becomes a sort of positive code, justification, and diagnosis in debugging [21]. This is particularly disturbing for complex software systems, where the cognitive load is created in grappling with the system, which might impair the capacity of developers with problem-solving and forming solutions.

2. **Psychological Characteristics:** Another key element in regard to human factors is taking into account people's psychological characteristics. Human beings are not all the same, but are endowed with difference personalities, motivation, and choices of preference. Such differences may therefore allude to the fact that how people collaborate with systems and other human beings is different in a grouped task environment. For example, a developer may enjoy working alone, while another performer enjoys working as part of a group. Better understanding of these differences and design, so that the development process can be much more effective and efficient [25].

3. **Social Features in Software Development:** For example, the social features of software development process also encompass the interactions among the team members. For example, understanding the communications and collaboration aspects and also the clarity of conflict resolution in a team of executing a software project can greatly influence results. For example, the team that communicates effectively and cooperates well can be able to work expeditiously and efficiently in staging a software of a higher quality [29].

2.4.1.2 Importance of Understanding Human Factors:

Understanding human factors, and ways in which they may comfortably be fitted in during the software development process, is no small matter. It demands a very multidisciplinary approach and cooperation in the sciences of computer science, psychology, sociology, and many others. While being hard, the benefits of human-centered software development can prove huge in improved team dynamics, higher software quality under development, and an increased satisfaction of the developers and end-users of the software [34].

2.4.2 Individual characteristics and cognitive abilities of software developers

Another breed of developers who have individual characteristics and cognitive abilities that affect their performance at work in a notable way is the software space's developing team. Software developers are definitely not entities who simply translate requirements into codes. They are, in other words, cognitive entities—certain characteristics, behaviors, and abilities that distinctly mark software development in the form of a product [30]. This set includes individual attributes such as cognitive style, personality, motivation, and cognitive abilities of software developers.

2.4.2.1 Cognitive Style

- **Definition:** Cognitive style is all that pertains to the way a person perceives, processes, and interprets information.
- **Importance:**
- Cognitive style of software developers assumes special importance in the fact that it lays an effect on the understanding or interpretation of the requirements and then on understanding the problem-solving decision-making process [25].

2.4.2.2 Personality traits

Following are the personality traits of the developers that show marked impacts on the process of software development.

1. Openness to experience
2. extraversion
3. emotional stability
4. conscientiousness

For instance, high conscientiousness among the developers would indicate organization and dependability, aspects very important in ensuring the development process is efficiently done and of high quality. In the same line, developers who have high openness to experience are, in most cases, very innovative, which is very essential in keeping up with the dynamics in software development [22].

2.4.2.3 Motivational Factors:

In the other form of individual characteristic defining software developers would be the motivational factors, for example, the intrinsic and extrinsic form.

- **Intrinsic motivation** could result in personal interest and enjoyment of the task, thus gaining higher creativity and problem-solving capabilities.
- **Extrinsic motivation**, on the other hand, may be outside rewards, among them compensation, which may have an impact on development performance and productivity [15].

2.4.2.4 Cognitive Abilities

Many software developers should have strong analytical and problem-solving skills for their cognitive abilities. Key cognitive abilities necessary for effective software development include:

1. **Analytical and Problem-Solving Skills:** These skills enable them to effectively analyze requirements, devise algorithms, and troubleshoot software issues.
2. **Spatial Skills:** Developers require spatial skills in the sense that they have to be capable of visualizing the software architecture and data structures
3. **Verbal Skills:** Developers also require verbal skills in team-based development of the software and communication with stakeholders on particular development plans [35]

2.4.2.5 Challenges and Considerations:

It is important to emphasize that even if these individual attributes and cognitive capabilities are largely important in software development, they are also potential sources of complexity. For example, the **clash of personality** between the developer of a different trait may result in conflict within the team, which will affect the performance of the team. Lastly, **different cognitive powers** foster differences in the distribution of tasks and imbalance of workloads, hence delays and inefficiencies [28].

This is to mean that the individual traits, cognitive ability, and the processing time of the software developers are the most critical factors in the software development process. In fact, such understanding of influential factors can provide valuable insight into approaches that developers take in carrying out tasks and acting in collaboration for contributing to the final product. All such concepts ultimately turn out as helpful in optimization of the process of development of the software and enhancing the quality of the product that software development companies present.

2.4.3 Psychological traits and their impact on decision-making

First, **"psychological traits"** are the individual differences in characteristic patterns of thinking, feeling, and behaving [41].
Another noteworthy point of interest, when learning the psychological aspect of software development, is that software developers have varied psychological traits and how these result in an effect on the decision-making processes. This is not only difficult to learn and understand, but of great importance.These are, as I have come to realize in the course of my study, the traits that will determine how an individual software developer is able to confront problems, interact with fellow team members, and the kind of decisions to reach in the process of software development.

2.4.3.1 Extroversion

The trait usually considered to belong to sociable and outgoing individuals, the target of many pieces of research, is the trait of extroversion. For instance, Cruz et al. [36] argue that extroversion is the trait of the software developer, which usually helps the team to be communicated via collaborative problem-solving. On the other side, an introverted developer would be likely to favor working alone most of the time and is likely most able on tasks that require deep focus and concentration [46].

2.4.3.2 Conscientiousness:

Other dimensions, such as conscientiousness, are said to show tendencies of organization, responsibility, and reliability. Research has shown that highly conscientious people are likely to produce more structured and well-thought-out decisions and contribute toward high-quality software products [53]. But, over-conscientiousness can lead to over-cautiousness of the individual and may result in delays for taking a particular decision [6].

2.4.3.3 Openness to Experience:

The last trait is openness to experience, which points to curiosity, creativity, and a preference for novelty and variability. Developers with high openness to experience are likely to search for innovative solutions that may fuel software development and product improvement [53]. However, if not managed, this trait could at the same time be a tendency toward overly complicated solutions or overly reaching the scope of the project [33]. The understanding of these psychological traits can plainly help the person in charge clearly handle software development teams. The personality of team members is huge in values to the project manager, who should take them into consideration when making decisions concerning the issues of the project.

2.4.3.4 Challenges

Knowledge of these factors can, in fact, facilitate the communication among the team in a healthy, harmonic working environment. This may, however, allude to high awareness levels of group dynamics by members of large organizations, a variable that may moderate. However, always do remember that the impact of the psychological traits on decision-making is a very complex and varied one, and that makes them sometimes with the potential effects that may be definitely beneficial or detrimental.

However, much can be learned from the dynamic interplay of these respective traits with each other, as well as the specific context of the project, the skills and abilities of team members, and external influences [45]. In essence, it is the psychological traits of a software developer that place a more central role in the decision-making process that happens during the development of software. The deeper knowledge of these traits can bring a better management style of the team, which contributes to better development of software and a better working environment.

2.4.4 The role of communication, collaboration, and teamwork

The role and real dimension of communication, collaboration, and teamwork for the development of software are real contributions toward effective project outcomes.

2.4.4.1 Communication:

The communication in this regard is to be mentioned that it is much more than a medium of information exchange. It allows participants to share their concepts, insights, and solutions, which encourages project progression [34]. This takes communication to be one of the basic ways through which mutual agreement among the team members is reached. In this context, proper communication in software development avails the transmission of a project's requirements and objectives to a worthy level in order not only to minimize any misunderstanding but also to minimize eventual deviations of the project [52]. It is also the decisive element in the solution of the conflicts and negotiations of the team. Kaur and Sharma (2020) would be realized to contribute to the productive addressable of any conflicts or disputes; thus, it would avoid potential project pitfalls [49].

2.4.4.2 Collaboration:

In the same light, collaboration denotes a situation of the interdependent working of the team. In essence, collaboration in software development takes some forms, such as pair programming and code reviewing, among others. Understandable, it is because of the fact that collaborative work environments offer multiple perspectives towards a single problem and hold a platform for learning and exchange of knowledge. Besides, software development is a complex work, and its development often calls for the integration of diverse technologies and expertise. Collaboration, if taken as a whole, is enabled to increase the problem-solving capacity of the team through the contribution of individual specialized knowledge and skill of members [52].

2.4.4.3 Teamwork:

Teamwork is integral in any form of project management, including software development. The necessity of teamwork is driven by the fact that together, results of higher quality and of more capability than those of isolated individual work are produced.

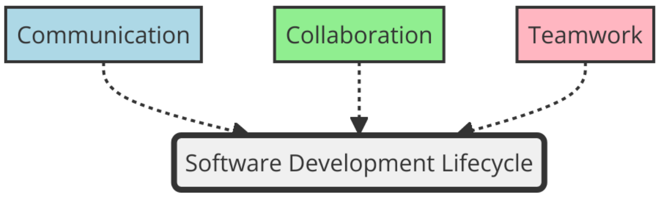


Figure 2.3: Key Social Factors Enhancing the Software Development Lifecycle: Communication, Collaboration, and Teamwork

2.4.4.4 Interrelation and Combined Influence:

On the other side, Alsaad et al. (2021) and Shafaat, and Qureshi (2020) have discussed how teamwork helps in adding value to enhancing the success of the project, whereby an effective problem-solving and increasing productivity outcome is due to the mix of different skills and perspectives [31,36].

In sum, communication, collaboration, and teamwork all have their separate contributions to make with respect to software development, and the net effect of all of them taken together is a manner of team working that is more wholesome and integrated, which contributes toward better results for the project. However, the actual contribution of these human factors should be recognized and integrated within software development projects effectively.

2.5 The Significance of Incorporating Human Factors in Software Development

In this regard, considering the integration of human factors in the development of software is something that we need to look after because of the large benefits gained due to this practice. Once reasonable attention is paid to human factors in software development, then only there is a possibility of having several major improvements in general [35].

The most visible benefit one can identify is **an improvement in team dynamics**. The consideration of the respective individual traits of each developer really helps managers orchestrate better teams. For example, a programmer with good problem-solving skills could work on some innovation to solve the bug, and another who is more creative could work on the design of the user interface [38]. In the studies of Görür et al., the teams oriented to the strengths of their members had very much better organization in the productivity and quality of output [25].

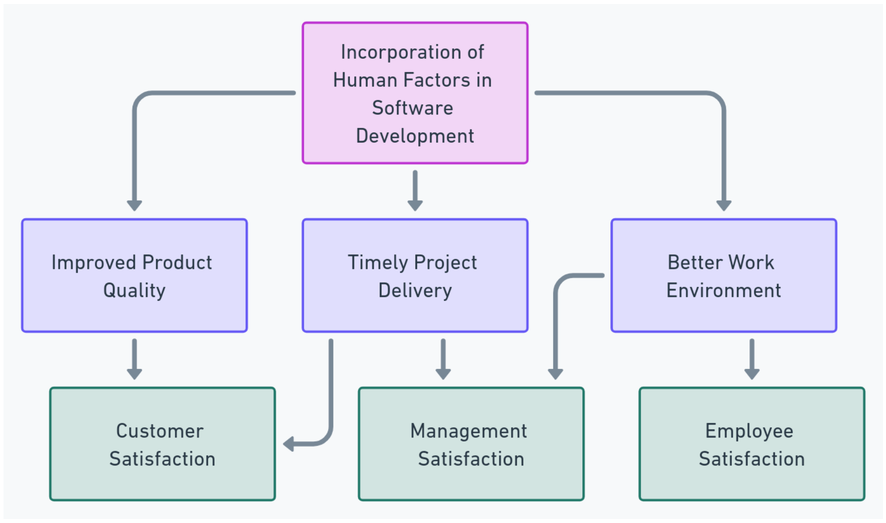


Figure 2.4: Flowchart demonstrating how incorporating human factors leads to enhanced satisfaction across different stakeholders in software development

Next, there is the subject of **stakeholder satisfaction**. It is evidently clear that software designed with high usability in mind, in consideration of the users' needs, will have a superior product meeting the requirements of the users. This is only possible through human-centered design practices, which refer to an approach whereby the needs and aspirations of end-users are understood and incorporated while designing the software. Sharma and Singh [39] had ascertained that the benefits of such practices were in user satisfaction.

Moreover, the role of human factors in improving the **quality of software products** should not be underestimated. "Developers who are more involved," meaning those who recognize the value of their own contributions and take their impact on end users' work seriously, produce better work [4]. This awareness of human errors in the field of software development ultimately leads to the use of robust procedures of error detection and correction, consequently improving the overall software quality [45].

Table 2.2. Table illustrating the diverse benefits of integrating human factors into software development, emphasizing the positive impacts on team dynamics, stakeholder satisfaction, software quality, development efficiency, and employee well-being.

Benefit	**Description**	**Impact**
Improved Team Dynamics	Enhanced communication and collaboration	Increased productivity and project success
Stakeholder Satisfaction	User-centered design leading to higher usability	Higher user adoption rates
Quality of Software Products	Attention to human errors and their mitigation	Fewer bugs and higher overall quality
Development Time and Costs	Efficient decision-making and problem-solving	Reduced time to market and lower development costs
Job Satisfaction	Acknowledgment of individual contributions	Lower turnover rates and higher team morale

Another considerable benefit is the **reduction in development time and associated costs**. Understanding the nature of this impact in development work, with the view of being put in a better position to make more informed decisions about design practices that most effectively limit the time and resources required to take it to the market [56]. In this study, Nguyen et al.1 found that, generally, teams who integrated human factors into their development process realized reductions in time and cost associated with software development.

The last advantage, often not considered when thinking about human factors, is **job satisfaction**. In a working place where the developer feels appreciated and sees some real outcome from his or her contribution, the level of job satisfaction will increase [57]. This is supported by a study done by Khairuddin et al. [64] in the year 2021, where he concludes that, indeed, employee job satisfaction does influence the retention of employees in companies that operate software whose value lies in human factors in work culture. It is very true that the above rationale demonstrates the fact that full realization of the said benefits can only come through deliberate efforts and sometimes structural changes in the approach toward software development.

2.5.1 Improved team dynamics

Improved team dynamics, directly related to embedding human factors in software development, is a complex topic that requires detailed investigation [33]. The better dynamics can be said to be improved patterns of interaction between team members. In essence, they are the development of collaboration, cooperation, or team building that results in a harmonious environment, which is effective and efficient [41].

2.5.1.1 Effective Communication

For sure, communication is a section of the very many dimensions. Communication is definitely one section in any team. Thus, misinterpretation, confusion, and definitely inefficiency will reign due to poor or even no communication. On the contrary, good communication enhances clarity, hence coordination, of which both are good recipes for the success of the whole team [49]. Normally, when the dynamics of the team are well enhanced, the communication channel usually is very open and effective because members feel free to air their thoughts and ideas [39].

2.5.1.2 Collaboration Enhancement

Another key area where dynamics can affect the functioning of a team is through the word "collaboration." Collaboration is active participation towards one goal from each of the members of the team [27]. A team that is capable of collaborating well is able to increase creativity when it comes to problem-solving or working effectively on complex problems. Effective collaboration often requires the balance of the skills, perspectives, and personalities that may best be enhanced through the incorporation of human factors in team management [66].

2.5.1.3 Psychological Safety

The psychological safety within a team also plays a crucial role in the team's dynamics. The group members in a psychologically safe team offer a climate in which they can take risks, be innovative, voice their ideas, and show their mistakes without the fear of retribution [56]. Thus, for the team, a psychologically safe environment could actually equate to an environment that supports innovation and learning by the team [62]. However, enhancing team dynamics does not come in a simple formula. Mostly, this often means to understand features of individuals and differences in cognitive ability together with cultural and personality differences.

2.5.1.4 Conflict Resolution

Managing these differences and making them aware of their presence might result in improved collaboration and cooperation within the team [26]. Besides, the other implication that human factors bring to team management is an improvement in strategies that pertain to managing conflicts within the team [58].

2.5.1.5 Ripple Effects on Software Development

It is may be worth noting, therefore, that improvements in these areas of team dynamics can ripple to improve other aspects of software development, including project management, quality assurance, and the final software product. Thus, incorporating the human factor in software development and the effect on team dynamic is indispensable.

In conclusion, incorporating human factors in software development has a profound effect on improving team dynamics. Better team dynamics enhance the communication, collaboration, psychological safety, and conflict resolution effectiveness between the team. This, in fact, is something that would create a very big positive effect on the way a software development project functions, in general.

2.5.2 Enhanced stakeholder satisfaction

We talk about the enhanced satisfaction of stakeholders in the backdrop of human factors in software development, which is really an issue of pressing relevance.

2.5.2.1 Diverse Stakeholder Expectations

According to stakeholders, the software development should involve customers, project teams, and the management and shareholders, each having different expectations and perspectives towards the software product.

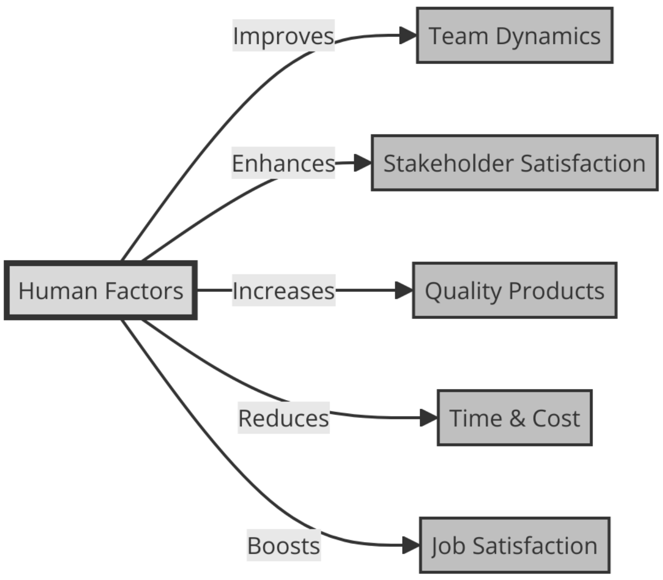


Figure 2.5: Impact of Human Factors on Software Development: A Visual Representation of Key Benefits

2.5.2.2 Quality and Communication

It would not be much of an exaggeration to say that if human factors are indeed part of software development, stakeholder satisfaction would be much improved. Hassan et al. claimed that even though this knowledge of developer individual characteristic and cognitive ability could enhance communication and collaboration within a team, it could then enhance the quality of the product [56]. Definitely, the accomplishment of high-quality products is bound to serve the interest of all the stakeholders, which includes, most importantly, the customers who are going to use the software and the management who would like to procure high levels of customer satisfaction.

2.5.2.3 Timely Project Delivery

It is also parallel to be noted that communication and effective collaboration in the team can also avoid any form of misunderstanding; hence, rework and delays stay at a minimum level. This will translate indirectly to being able to deliver the project within the stipulated time frame, one of the critical indicators in successful project management, thus satisfying both the management and the customers anxiously waiting for the software. This is also affirmed by Sáez et al. in their research of the year 2021, which reveals that the level of satisfaction of the stakeholders increases notably through the delivery of projects on time [28].

2.5.2.4 Work Environment and Dynamics

In human factors, this may lead to improved group dynamics in an environment that works well [36]. This is because it improves group dynamics to an environment that works well. This will increase job satisfaction and employee retention, hence satisfying the internal stakeholders' needs.

2.5.2.5 Informed Decision-Making

The human influence does not remain limited to it: perspective of human factor can enable project managers to make most of the informed decisions through keeping in view the psychological traits and cognitive abilities of the team members. This will, by extension, result in more effective resource use, cost reduction, and development time [45]. These results in increased return on investment (ROI), hence would clearly lead to satisfying one of the main stakeholders to any project: the shareholders.

Apparently, the inclusion of human factors in software development could mean better satisfaction of the stakeholders in a number of different ways. The benefit of considering human factors ranges from improved product quality, on-time delivery of the project to ROI, and creating a better working environment. However, detailed empirical research is required in this area so that these arguments can be further confirmed with more concrete evidence.

2.5.3 Higher quality software products

From a software quality perspective, human factors are highly important to the development process. The proper introduction of human factors in the software development environment can produce high-quality products.

2.5.3.1 Human-Centric Development:

- **Premise:** On the other hand, the human factor has been put at the center of the software development cycle. Software development is a human-centric activity. The development aims at a human being's way of work, minimizing the chances of error. In the software development cycle, a human plays the central role. An individual who well understands the problem designs and implements the All those methodologies—be it Agile, Waterfall, or Scrum—put prime focus on the human interactive role and understanding for delivering quality software [54].
- **Outcome:** The contribution the human factor makes, when integrated into the software development life cycle, is that the product that gets developed shows an

amazing improvement in quality levels. The quality of the software is affected by human factors in a couple of ways. First, when software developers work in a creative fostering environment, there is the highest possibility that they could come up with new solutions and innovations, and these are definite factors toward the higher quality of software.

2.5.3.2 Communication and Understanding:

Secondly, the human factor-inclusive development environment does tend to promote good communication in the development team and with stakeholders. Good communication practices make understanding of software requirements easy, which is one of the important aspects of software quality [89]. Furthermore, the environment encourages users to be able to provide their feedback for consideration in the further development of the software.

Table 2.3. Table outlining strategic implementations of human factors in software development to produce higher quality software products.

Strategy	**Implementation**	**Outcome**
Human-Centric Development	Focusing on user needs and error minimization	Products that meet user requirements with fewer defects
Diversity in Problem-Solving	Leveraging varied developer backgrounds	Innovative solutions and comprehensive problem addressing
Continuous Feedback Loop	Encouraging user and team feedback	Iterative improvements and user satisfaction

2.5.3.3 Diversity and Problem-Solving:

It is expected that diversity among the background and experience of software developers will bring greater anticipation and avoidance of problems in software design and development. The expected confluence of diversity should result in more software products free of bugs and letting software fail less, which is a sign of good-quality software [80].

2.5.3.4 Strategic Implementation:

- **Best Practices:** However, it is only when the human factors are put into perspective during the development of the software that it results in a quality software product not necessarily always possible. The way one does it is of paramount effectiveness and efficiency, and this involves developing proper strategies and best practices that would improve team dynamics, promote communication, and encourage creativity [55].

- **Supportive Environment:** One such practice includes developing a positive work environment that ensures an appreciation for the contribution from each member of the team. This may imply that their success is recognized and availed with opportunities for learning and growth, along with a culture of respect and trust [8].
- **Feedback Mechanism:** The other practice is the regularity of feedback and reflection. This can help the team know any problem or challenge present in the development process, and it comes up with effective solutions [67].

This means, therefore, that the software development exercise may result in a high-quality software product that involves the human factor. On the other hand, it would require being put in place strategically and effective implementation in an appropriate practice and the environment that supports the developers of the software

2.5.4 Reduced development time and cost savings

Time factor into the software development is commonly known as a very time-consuming process. Certain aspects might directly influence this factor, like human factor. In doing so, consideration and integration of human elements in software development may, in fact, bring about a case for reduced development time and some cost savings of several million dollars, bringing about efficiency in operations and profitability. This has been substantiated by several recent studies [35]-[37].

2.5.4.1 Error Reduction and Efficient Task Allocation:

- **Minimized Errors:** Reducing the number of errors during software development is the most tangible way that human factors help save time in the development process. Studies showed that developers with the given personality traits tended to make fewer mistakes [45]. Therefore, through error reduction, it means one has significantly reduced the time used in debugging and reworking, thus ensuring that the process of development is efficient.
- **Strategic Task Distribution:** Better understanding and management of the human factors may assist in improving task allocation, where tasks are given to the most suitable developer regarding his skills, interests, and cognitive abilities [58]. This will help in completing the projects in a timelier manner.

2.5.4.2 Enhanced Communication and Collaboration:

- **Streamlined Communication:** These two human-factor concepts play a critical role in reducing development time and both take the high stage and cannot be underscored. On the other hand, poor communication most of the time will culminate in misunderstandings, errors, and hence the delay in the process of development.

- **Collaborative Synergy:** In addition, smooth collaboration involves the quick transfer of knowledge and problem solving among team members, which further helps reduce the consumption of development time [67].

2.5.4.3 Cost Reduction through Human-Centric Approaches:

- **Lower Turnover Rates:** The advantages have to do with everything regarding savings on cost. Of course, reduced development time carries into decreased costs, but there are other dimensions as well. For example, research on human factors might lead to job satisfaction, which could work in their favor to lower their turnover rates.

- **Improved Workplace Health:** Additionally, better understanding of human factors may lead to a healthier work environment, thus fewer sick leaves and more productivity again contribute to cost savings [69].

2.5.4.4 Challenges and Future Directions:

However, it is to be said that all this is not easy to realize. This goes on to show that understanding and effective management of human factor require an investment; besides, the impacts may vary within the environment. These need to be further examined in future studies to devise more concrete strategies and practices [52]. However, the human-factors perspective in software development can evoke some important gain in the process that makes the relevant field of importance. The potential for reduced development time and cost savings certainly provides a strong case for its consideration.

2.5.5 Increased job satisfaction and employee retention

Human Factors will have very important results in increasing the satisfaction in work and the retention of employees, and that is introducing the human factor in software development. These both things are interconnected and basically feed each other, making it a virtuous cycle within the software development environment.

2.5.5.1 Job Satisfaction: The Core of Productivity

Satisfaction from work is a term with which it would be safe to say how satisfied one is with his job. In other words, it is a proportion of the number of people who are affected in a positive way or a positive attitude toward the job [45]. Nature of the work setting, styles of leadership, individual differences and personality traits, and work values—all do have their impact on job satisfaction [65]. Contextually in software development, the chance of being satisfied at one's workplace is most influenced by independence, recognition at work, professional growth, and good communication, and is not expressible independently. If, in truth, the human factors were considered in software development, it would mean that most likely the developers will feel much appreciated and understood by their environment. Looking at the developers as unique individuals, possessing their strengths, cognitive abilities, and ways of doing work, it will only add up to inclusivity at work alongside empathy. These would, in turn, lead to increased satisfaction among the developers since each one of them would feel respected, and their personal needs, ideas, and views would be taken into account. A study conducted by Aziz et al. identified that work environment, teamwork, and recognition of work have a significant positive relation with job satisfaction among software developers [57].

2.5.5.2 The Ripple Effect on Employee Retention

Increased job satisfaction among employees thus also contributes as a byproduct toward increased employee retention in the organization. Employee retention is a measure of an organization's ability to retain its employees over a given period [66]. This is especially important in the software industry because of the very high costs of searching for and training new employees, coupled with the potential loss of good institutional knowledge when employees leave the organization [35]. When job satisfaction exists, a software developer is less likely to be searching for another job and thus contributes to retention.

2.5.5.3 Strategies for Maximizing Job Satisfaction and Retention

- **Recognizing Individuality:** Human factors integrated in software development could increase job satisfaction among developers and even create a sense of organizational loyalty and belongingness for their organization.

- **Fostering Growth and Development:** These are valued and well-appreciated, growing opportunities and the development of software that would provide them with the environment to work positively in collaboration, thus keeping the organization motivated for the long stickiness.

- **Enhancing Work Environment:** Adding human factors to software development might make a great positive difference in job satisfaction and thereby the retention of the employee.

In this respect, then, it implies that organizations can only be in a position of ensuring that inclusivity and empathy are good enough to increase job satisfaction and for holding onto employees by recognizing and regarding strengths, needs, and perspectives of the software developers themselves as individuals.

2.6 The Challenges of Incorporating Human Factors in Software Development

While including human factors in the software development is deemed to be beneficial, various challenges tend to rear their ugly head when including these very human factors. These are

- The need for more investment in staff training
- The complexity of management witnessed in diverse teams
- The challenge of seeking balance between well-being and production
- The confronting of the negative effects that are cognitive biases and decision-making errors

The major challenge is the training of personnel. Examples of human factors integrated into software development include implementing training programs that will make sure the entire staff is on board with new procedures and protocols [43]. They may also take much time and financial investment on the part of the organization. In that case, an organization would have to take into consideration the cost of these training programs and the extent to which such training programs could ideally enhance the overall process of developing the software.

Diversity team management is an additional challenge in a highly diverse virtual workplace. Sometimes, this will result in diverse teams that cause innovative solutions for problems and bring about a more inclusive work environment [244]. This becomes a complicated task with a tendency that diverse teams tend to suffer from conflict, misunderstanding, and the decrease in team cohesion. These potential negative implications would affect very largely the success of the software development project and therefore need to be managed proactively [44].

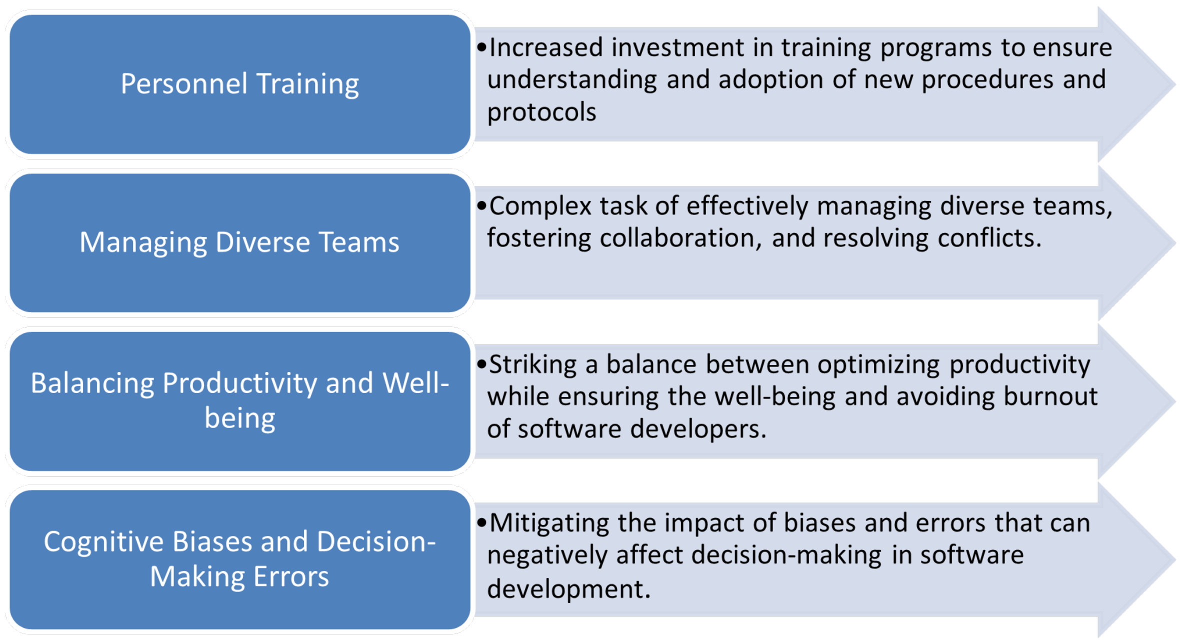

Personnel Training
•Increased investment in training programs to ensure understanding and adoption of new procedures and protocols
Managing Diverse Teams
•Complex task of effectively managing diverse teams, fostering collaboration, and resolving conflicts.
Balancing Productivity and Well-being
•Striking a balance between optimizing productivity while ensuring the well-being and avoiding burnout of software developers.
Cognitive Biases and Decision-Making Errors
•Mitigating the impact of biases and errors that can negatively affect decision-making in software development.

Figure 2.6: Challenges in Incorporating Human Factors into Software Development

Another challenge of integrating human factors into software development is that of balancing production and quality of life. While paying attention to human factors could result in better productivity of employees and job satisfaction, there is also a possibility that it may They equally have the possibility of preventing any risk that comes with it. Striking the balance often becomes too hard, especially in high-pressure environments, such as software development.

Finally, another big difficulty that comes in incorporating human factors into software development is cognitive biases and their errors in decision making. Such biases and errors greatly compromise the quality of the decision-making process within software development and thus directly lead to suboptimal solutions and outcomes. Therefore, organizations should be aware of these and take required measures in mitigating the impacts of such issues.

The potential gains that would be realized by incorporating human factors in software development cannot be underscored. Such challenges, therefore, have to be factored into the overall planning and implementation of strategies to incorporate human factors in the software development processes of the organization. However, much further exploration is needed into how to handle it [61]. Challenges need to be considered in very attentive manners when one tries to incorporate human factors in software development processes so that possible benefits are realized.

2.6.1 Increased investments in personnel training

Inclusion of human factors in the development of software calls for an important commitment to knowledge and development of human potential. It is an investment to be put in place and demands to be dedicated with substantial resources, particularly towards personnel training [45].

2.6.1.1 The Imperative of Continuous Learning

- **Dynamics of Human Potential:** Such investment is based on the logic—human element is never static within the system; he is a dynamic entity, which is subjected to growth, learning, and evolution.
- **Impact of Effective Training:** Research shows that good training for personnel vastly increases the performance of the team and better communication, thus having a high quality of produced software [54].

2.6.1.2 Navigating the Cost Complexities

However, training is costly, time-consuming, and a complex process that demands considerable financial and time investment. This is, therefore, mainly due to the various learning styles and tastes that people have. Sixty-six percent of them opined that the learning approach needs to be personalistic in order to achieve effective learning outcomes [66]. The designing, implementing, and maintaining of a productive training program, therefore, cost something dear. Personnel training costs are so much more than money costs. However, this time that will be used in such training activities is most likely going to be seen as a loss of good time for actual software development, hence resulting in delayed project timelines [58]. Moreover, given the fast pace at which the tech ecosystem evolves, training content will need to be recurrently updated in order to remain valid, i.e., repetition of investment over time.

2.6.1.3 Maximizing Return on Investment

Indeed, a good training program has to be very agile and constantly on the move in order to soak in the new methodologies, tools, and concepts—a dynamic, regular process rather than a static one-off event. Broad investments in personnel training, if conducted strategically, may bring positive returns—even if this, in a common dimension, remains unnoticed due to their perceived costs. This would further improve the productivity and efficiency of the team through training programs on core software development skills, collaborative skills, effective communication, and problem-solving [56]. This, in the long run, nurtures professional development among the developers, leading to better job satisfaction and reducing levels of turnover [86].

This is to say that it may pay off the cost of training staff with quality improvement of the software product and productivity of the development team. To some extent, the "human factor" on the path of software development, primarily staff training, is quite an investment. The gains that may accrue from the investment, however, could easily outweigh the costs if properly managed. It needs to be a balancing act between the need for personnel training against costs that are incurred. Further research would be needed in order to actually evaluate this cost-benefit ratio with regard to the incorporation of human factors—in the form of personnel training—into software development.

2.6.2 The complexity of managing diverse teams

This has also been constant with time, whereby software development has improved to be a globally distributed work. In this respect, the increasing more integral part of the process has been taking place with diverse teams [66].

2.6.2.1 Communication Barriers and Cultural Diversity

Cultural, gender, ethnic, skills, experience, or even personality traits differences. This adds significantly to the challenge of managing diverse teams such as these.

- **Language Barriers:** One of the greatest challenges in software development with diverse teams is related to communication [59], whereby due to language, culture, and time zone diversity, it can be a real blockade to sharing ideas and information. For example, in a language barrier, the ideas that will be clear and precise will definitely not be communicated; hence misinterpretation.

- **Cultural Differences:** Others are cultural differences; it may come in different forms of social etiquette's and norms that sometimes work against the full understanding and respect of team members—misunderstanding or conflict among team members [55].

- **Work Ethics and Practices:** In addition, the different work ethics and practices may pose challenges [58]. Team members belong to varied cultural backgrounds, thus having diverse work ethics, varied productivity standards, ways of executing tasks, etc. This can be a cause for friction and even misunderstanding among members, therefore affecting the general work output.

- **Time Zone Differences:** Last but not least, the big barrier arises from the time zone differences when trying to coordinate a synchronous communication or a meeting [59]. One more complication would be dealing with managing conflicts that can be coming out of diversity, which is possibly arising from any point of views due to a different nature or even from bias and prejudice.

2.6.2.2 Managing Conflicts Arising from Diversity

With improper management of these conflicts, the team cohesion is influenced, thus affecting the performance of the team [76].

2.6.2.3 The Advantages of Diversity

Despite these complexities, there are significant advantages of diverse teams.

- **Enhanced Creativity and Problem-Solving:** A diverse team can pool the different skills, knowledge, and experiences they hold, fostering creativity and problem-solving.
- **Increased Flexibility:** This could help in giving different perspectives that will bring better decisions and innovative solutions. This is also likely to increase the flexibility of the team towards various situations [77].

2.6.2.4 Strategies for Effective Management of Diverse Teams

1. **Creating an Inclusive Environment:** The management, therefore, in this software development basically has a challenge of really managing those complexities well so that they enjoy the advantages that diversity brought about by the teams brings. This may involve creating an environment of inclusively, where every member of the team feels valued and, in turn, his or her inputs appreciated.
2. **Investment in Training:** Training may be something of necessity should improvements in communication and collaboration among team members be achieved.
3. **Timely and Effective Conflict Resolution:** Further, timely and effective mechanisms of conflict resolution need to be in place to the latter [67].

Such are some of the difficult ends to manage in software development. The benefits of having diverse teams in creativity, problem-solving, and adaptability greatly outweigh those of uni dimensional teams, provided the challenges encountered are managed appropriately.

2.6.3 Balancing productivity and well-being

Further, it is a challenge to ensure a balance in productivity and well-being, at the same time incorporating human factors in software development. The competitive nature of the software industry raises much consequence in today's working environment. The productivity level within the organization, with regard to software development, had traditionally been considered against the number of high-quality software produced by the organization in a given period [58].

2.6.3.1 The Interplay Between Productivity and Well-being

- **Employee Well-being:** Employee well-being is one of the important elements for productivity at both physical and psychological levels; it includes satisfaction, health, level of stress, and mental health. Stress, burnout, and long working hours are very common in the software development industry.
- **Conflicting Variables:** From recent studies, there is a very strong positive relationship between the well-being of the software developer and the quality of software produced [62]. That means here is a balance that has to be done between software developer productivity and their well-being to get the desired output. In this software development scenario, at times, these two variables come up with some sort of conflict. Time pressure has at times led to the forcing of employees to work beyond what they can afford. This has resulted in stress and ultimately to counterproductive well-being. Thus, there is a balance, and this, as everywhere, there are controversies. High workload and time pressure may affect the well-being of developers to the extent that the consequences at the individual and organizational level are desired. Over time, the employees then develop high levels of stress when they are overloaded with the work, or the work is characterized by a lot of pressure at different times, hence sometimes pushing the process of burnout further and lowering the level of job satisfaction [74].
- **Crunch Time Culture:** In software development, the culture of crunch time has been noted, whereby developers are put to work enormously long hours to meet the tight project deadlines. Though these elements of work can lead to enhancement of short-term productivity, in the long run, these would most likely result in burnout and low job satisfaction, which eventually facilitates intention to leave [65]. In fact, a recent study found a large negative correlation between long working hours and high job satisfaction among software developers [58].

2.6.3.2 Strategies for Balancing Productivity and Well-being

A concern should be balancing employees' working hours, satisfaction with the job, and health for any given productive line of work.

- **Flexible Working Hours:** Flexible working hours will improve work-life balance for the developers and consequently improve job satisfaction and reduce stress levels [86].
- **Meaningful Work and Positive Feedback:** Some other factors that could be enhanced in job satisfaction may be meaningful work, positive feedback, and opportunities for professional development [73].
- **Promoting Physical Health:** Besides, the institution may contribute to the health of its employees, with efforts including the physical exercise carried out at work and the healthy food offered to employees at their workplaces.

Balancing between productivity and health of the workforce in software development is really tricky, for which one needs to have a comprehensive understanding of the human factors involved. This should very clearly indicate that indeed organizations should ensure that the overall well-being of their developers is improved, taken care of, and consolidated for very high levels of productivity to be sustained in the long run. Future research shall be directed at deriving ways in which improved productivity and well-being are attained and sustained.

2.6.4 The impact of cognitive biases and decision-making errors

In the software development context, cognitive biases and the resulting decision errors could lead to impacts of a more widespread nature. Cognitive bias refers to any systematic deviation from rationality in judgment and decision-making [81]. They are "shortcuts" inborn in the human psyche that develop over time for efficient decision-making, especially in complex situations. Sometimes, such biases led to the misperception or interpretation of something else or an irrational decision [83].

2.6.4.1 Cognitive Biases in Software Development:

1. **Confirmation Bias:** One of the major ones that tend to lead to poor judgment and affect decision-making within software development is the confirmation bias. The selective favoring of information seeking, which confirms preconceptional beliefs or values and disregards evidential contradiction to its content [72]. Thus, within the context of software development, when developers prefer solutions or strategies of which they know or like, even in the presence of situations that present other solutions as more effective or efficient, that is thus indicative of a situation of confirmation bias [63]. This can lead to overuse of some technologies, even when these are not fitting the best way to solve a problem.

2. **Overconfidence Bias:** The other most common is overconfidence bias, in which a person is overconfident of predicting something. Overconfidence can lead to misjudgment in project timelines and budgets; it highly influences project management, due to which deadlines are not met, and resources are overuse [63]. For example, the developers could underestimate the time of completion of a given task. This, therefore, shows overconfidence in doing something that is getting done, thus leading to delays in projects.

3. **Anchoring Bias:** Another cognitive bias that might set in through decision-making in software development is anchoring bias. Anchoring is a kind of bias in which people place too much weight on the first piece of information that comes to their attention (the "anchor") when making judgments or decisions [54]. First, it would increase the likelihood of holding onto the initial estimates or decisions made, even though the new information might make this point the opposite way.

4. **Groupthink:** A phenomenon in which a group makes faulty decisions because of group pressures that lead to a deterioration of mental efficiency, reality testing, and moral judgment [56]. Groupthink may hamper individual creativity that is likely to show a new way for problem-solving, with a combined impact on the quality of software development.

In all, the developers always have to make a decision in respect of the design, implementation, testing, and others. A lack of it or cognitive biases may bring about faulty interpretations of the information [67]. In sum, the procedures of software development are much affected by cognitive biases and errors in decision-making. From project management to software design and implementation to testing, it has a great role in all. It is upon this significance, therefore, that there is a need to understand these biases and decision-making errors to control them effectively and hence make the software development process efficient and successful.

2.7 Cost-Benefit Analysis in Software Development

The area of software development has largely been dynamic over the years, whereby a huge variety of variables influences the overall success of the project of development. A Cost-Benefit Analysis (CBA) is hence a very important and imperative technique so that these varieties take account, evaluate choices, and reach appropriate decisions. CBA is actually an economic technique using systematic calculation to compare the costs against the benefits of a project or decision [99].

Table 2.4. Table presenting the reasons why conducting a cost-benefit analysis is crucial for understanding the implications of incorporating human factors in software development projects.

Justification	**Explanation**	**Expected Benefit**
Economic Feasibility	To ascertain the economic viability of integrating human factors	Helps in decision-making regarding investments in human-centered practices
ROI Determination	Understanding the return on investment for human factors initiatives	Provides a basis for allocating resources efficiently
Long-Term Planning	Evaluating the sustainability of human-centered development practices	Assists in formulating strategies that ensure long-term success and employee well-being

2.7.0.1 Costs and Benefits of Incorporating Human Factors

More concretely related to software development, CBA should help understand trade-offs between different project strategies and options by comparing the prospective costs and related benefits of the proposed software system [100]. In respect to this, the results of CBA can be used in making decisions of either guiding some decision or all of them, e.g., the judgment of the feasibility of the project, the most efficient development methodology to be used, or the optimal way to allocate the resources.

- **Costs Considerations:** Costs in the line of software development usually include tangible infrastructure personnel and the software or hardware tools, and there are also intangible costs, including the project risk, time, and opportunity costs of the alternative investments [103].
- **Benefits Realization:** Benefits, on the other hand, are normally harder to quantify and may include tangible benefits such as the expected revenues, accrual benefits, and intangible benefits like better efficiency, improved customer satisfaction, and strategic value for the organization.

However, the emphasis here should be that CBA is not a "crystal ball" that can make exact predictions of the future outcome; rather, the tool gives an estimate based on available data and under a certain set of assumptions. In short, the reliability of CBA solely depends on the quality of the data used and the assumptions made [104].

2.7.0.2 Decision-making Insights:

Besides, it functions as a powerful tool in the evaluation of the financial feasibility of accounting for the human factor in software development. The consideration of human factors may be complex and bring in requirements to the project for more resources like building a team, training, and sometimes even psychological support. These investments, in turn, might lead to handsome dividends in the long run, such as better team dynamics, improved job satisfaction, and, eventually, higher-quality software products [105].

These understandings of costs and benefits of human factors provide some basis on which organizations might base their software development strategies and make more rational choices. For example, net benefit "positive" shows that consideration of human factors brings an advantage higher than the cost, and hence would indicate signs of bearing fruits as a result of investing in it. When the net benefit is actually negative, this may imply that the costs outweigh the benefits, and therefore the strategy ought to be re-evaluated [106]. The sections that follow critically discuss the concept of CBA, its application in the software development process, factors which affect the costs and benefits when incorporated into human factors, and finally, some examples of cost-benefit analyses in related fields.

2.7.1 Introduction to cost-benefit analysis

Cost-benefit analysis (CBA) is a very large field and represents a systemic approach for one to calculate and compare costs and benefits in a project or decision [101]. In principle, this is very highly used in economics to assess the efficiency of given interventions. Economic efficiency theory postulates that the benefits of an intervention are worth undertaking if the sum total of the intervention will be greater than the remitted sums of money.

2.7.1.1 The Essence of Cost-Benefit Analysis

The foundation of cost-benefit analysis in software development is grounded in the basic utility theory of microeconomics. "In the software development field, CBA makes the decision-makers see clearly the potential economic impacts of different decisions" [103]. The purpose of CBA in software development should be to analyze whether, and if yes, to what degree, a certain decision regarding software development or the developed project itself is economically justifiable. It provides a numerical value, hence very useful, as it helps in comparison of various decisions that may affect the software development project and hence selects one that is most efficient among them.

2.7.1.2 Methodology and Challenges

- **Simplification and Comparison:** The basic idea of CBA is rather simple: all costs and benefits related to any kind of activity or decision are brought to a common measure, in most cases, to monetary units, which is usually done for the ease of comparison. The costs usually include direct costs, for example, expenses for the development tool or training of developers, and indirect opportunity costs. Benefits generally include revenue from the project or decision and saving through an increase in efficiency [104].

- **Quantification of Intangibles:** But then, despite the apparent simplicity, application of CBA can be rather daunting. All the relevant costs and benefits have to be identified, numbers have to be put against them, and reduced to present values—all these are subjective judgments, besides being affected by the availability and reliability of data. Some forms of benefits and costs, mainly those that are intangible or unpredictable, may quite pose a challenge in being quantified into monetary units. For example, when trying to quantify benefits such as improved collaboration among teams or that of the developer's quantifiable satisfaction in a team, it becomes hard. Likewise, the potential costs of project failure or delays may not be readily predictable.

2.7.1.3 Criticisms and Utility

- **Critique of Simplification:** On a general scale, CBA has also not been free from criticism. Some of the critics are of the notion that it only tries to simplify very complex decisions and does not take into account qualitative factors. They also argue that it is likely to undervalue or ignore benefits and costs that would find no ready monetization [105].

- **Value in Decision-Making:** However, CBA still remains a very useful tool in software development. It supports rational decision-making by systematically framing a viewpoint on quantitatively weighing the economic pros and cons of different decisions. This gives an informed choice for the project managers, project developers, and any other stakeholders based on economic efficiency, which eventually contributes to the success of the software development project.

This section explains the concept of a cost-benefit analysis, its methods, and the application in software development; the respective challenges are given for the same.

2.7.2 Applications of cost-benefit analysis in software development

Software development is a field characterized by high complexity, full of such complications that can be solved effectively and their solutions applied in the development of software, hence completed with success.

Cost-benefit analysis is a systematic approach within the software development field, used to estimate the strengths and weaknesses with the alternatives to determine the options that best provide an approach towards benefits achievement while saving. It focuses on appraising the total costs, benefits, and other impacts to the stakeholders, directly and indirectly, involved in any project [109].

Anyway, such software development initiatives are not an island of coding themselves but rather part of a wide array of activities, including requirements gathering, system design, programming, testing, and maintenance [104]. With it, they are accompanied by costs that would be able to quantify themselves in financial metrics under the respective circumstances. On the other side, some of the advantages may include better operational efficiency, customer satisfaction, and a good ability to compete, gaining bigger market share with lower costs of maintenance, among others.

2.7.2.1 Assessing Project Viability and Methodology Selection

- **Feasibility Studies:** The cost-benefit analysis may be useful in many applications as a software development life cycle initiation. The feasibility study, which is meant for establishing the viability of the project and if it's really worth taking, involves carrying out an activity. One of the major components of the feasibility study is the cost-benefit analysis, which helps in ascertaining if the given software project should be economically beneficial [106].
- **Methodology Selection:** Moreover, it places added essence on the decision-making that is related to the choice of the software development methodologies through the application of cost-benefit analysis. A well-established practice within development methodologies is fast growing, given that the utilization of any of the methodologies, having their strengths and weaknesses, has to be identified for the most suitable use within a given project [107]. In this way, it can be done to carry out a cost-benefit analysis with other methodologies, like Agile, Waterfall, DevOps, among others, taking into consideration their financial impacts, time to market, quality, and all other differences associated [108].

2.7.2.2 Guiding Software Maintenance Decisions

The second most important application area is during software maintenance with cost-benefit analysis. This may form a greater percentage of the total life-cycle costs [109]. The cost-benefit analysis applies when a decision to continue with maintenance of the software product, to consider a major upgrade of the software product, or to discontinue its production is necessary [110].

To sum it up, the applications of cost-benefit analysis in software development are too many and spread across many aspects of the software development life cycle. Therefore, this becomes a decision tool that the project managers and any other decision-makers can utilize in availing themselves of information that may be useful to them in making a decision on a course of action or even simple decisions that may be very important in determining the success of the software projects [111].

2.7.3 Factors influencing the costs and benefits of incorporating human factors

The human factor integrated with the software development approach brings a number of influences in determining the costs and benefits involved. It is prime to understand that these influences deployed and controlled in the development ecosystem can therefore bear fruits either positively or adversely.

2.7.3.1 Training and Development Costs

The main factor involves the cost that is required during training and development of the employees. Whenever the human factor is applied in the development of software, the skills and other capabilities should be enriched from time to time, based on the knowledge base of the concerned employees [112]. Though it increases productivity, enhances quality, and increases creativity by the employees being trained, this may be very expensive. Further, there is a time cost, which is training taken from doing productive work. Thus, an investment in training can have both cost implications and potential benefits [112].

2.7.3.2 Organizational Culture and Communication

The other key aspect is the cost that comes about in organizational culture change and that of communication patterns. Team communication and collaboration form part and parcel of human factors involvement [3]. It would, however, be a costly and time-consuming affair to try to change the communication and organizational culture. The increased productivity, less error rates, and better behaviors of working together in teams will be among other potential benefits that can be outweighed.

2.7.3.3 Psychological Traits of Software Developers

Another important factor that determines cost and benefit lies in the individual psychological traits of software developers. For instance, resistance to change or lack of collaboration for some traits may negatively affect team productivity, where high resilience and adaptability to change with high learning orientation might mean increased productivity and innovative solutions [1115]. However, there is a development of systems and procedures that can effectively manage these individual differences and can add to the operational cost.

2.7.3.4 Technological Infrastructure

Further, the support technological infrastructure for the incorporation of human factors in software development can also influence cost and benefit. Other tools that could go a long way in ensuring software development that is in line with human-centric approaches would include, among others, project management tools, virtual collaboration tools, and time-tracking tools. However, the acquisition and maintenance of such technological infrastructure also incurs cost [116].

There are many factors that affect the costs and benefits of integrating human factors in software development. Realizing the balance between these will help in maximizing benefits with minimal costs accrued.

2.7.4 Examples of cost-benefit analyses in related fields

The employment of human factors in software development is, therefore, a delicate task. It involves huge arrays of advantages but bears some costs. There are many factors that affect the cost and benefit; hence, they also affect the performance and efficiency of the process. Such causes would primarily entail individual characteristics of software developers, characteristics of the software development projects, and the overall organizational context.

2.7.4.1 Impact of Individual Characteristics

Key factors contributing to it are the cognitive abilities and personality traits of the developers. For example, high cognitive developers may be efficient and effective in their tasks, bringing in more productivity, hence benefits. However, a less agreeable nature means lower productivity, and costs will have to be paid in such cases by the developers. This highlights a point of individual differences, which must be considered during incorporation of human factors into software development [120].

2.7.4.2 Project Characteristics

What type of software development project it is also has a role in costs and benefits. For example, highly specialized knowledge for complex projects should benefit more with the employment of human factors that enable effective ways of collaboration and problem-solving. In contrast, for the simpler projects, the investment of time and resources required in training and development might be greater than the payoff for project success.

2.7.4.3 Organizational Context

The other critical parameter is the organizational context. Human factors tend to be well integrated when there is supporting organizational culture and resources within reach; hence, the benefits are prone to increase. An organization less supported by resources and culture would then lose out, as they will incur more cost for incorporating human factors [114].

2.7.4.4 Methodological Influence

Further, the methodologies used are in a position to influence the costs and benefits. In this scope, agile methodologies, based on collaboration and individual interaction, for example, are the ones that could rise to the benefits of the integration of human factors [115]. Considering the produced resistance to change, on the other hand, the more traditional, inflexible methodologies are likely to increase the costs.

2.7.4.5 Socio-Economic Considerations

On this point, the overall necessity also arises in relation to considering socio-economic conditions. They are comprised of the local conditions prevailing in the labor market, the level of technological advancement, and the regulatory environment which is likely to bear an impact on the costs and benefits. For example, areas with greater competition for human capital are likely to have benefits arising from the application of human factors to the working environment, which may be appreciably high.

In sum, the costs and benefits for the integration of human factors into software development are many and include the following. Therefore, such issues need to be taken into consideration by the organization so that the accrual of maximum benefit takes place and costs are at minimum levels before deciding to carry out the integration of the human factor into software development. Future research would continue to examine such relationships and interactions to better nuance the cost-benefit analysis of incorporating human factors into software development.

2.8 Gaps in the Literature

This review clearly shows that there are gaps in the existing body of knowledge in cost-benefit analysis of the incorporation of human factors into software development. Very few among such individual detail research on the topic is an amalgamation of these different ideologies and some sort of all-inclusive analysis of the same.

Table 2.5. Table outlining critical gaps in the current literature regarding the incorporation of human factors in software development, suggesting areas for future research.

Research Area	**Gap Identified**	**Suggested Focus for Future Research**
Economic Impact Analysis	Lack of detailed cost-benefit analysis	Comprehensive studies evaluating the ROI of incorporating human factors
Long-Term Effects	Limited insight into long-term impacts	Research on sustainable practices and their long-term benefits and costs
Methodological Diversity	Predominance of qualitative over quantitative studies	Implementation of mixed-methods approaches for a broader perspective
Cultural and Organizational Context	Focus on specific contexts limiting generalizability	Cross-cultural and diverse organizational studies to widen applicability

2.8.0.1 Lack of Explicit Cost-Benefit Analysis

- **Observation:** As reviewed in the relevance of human factors to software development, very many works have dealt with this relevance, yet very few have ventured into an explicit cost-benefit analysis in this regard [121]. Existing literature indeed acknowledges that human factors should be part of software development, but it does not delve further into the details on how the financial implications for software development can be determined.
- **Implication:** This should actually make such an investigation into the financial impact of such integration, not in a esuriant way, yet not merely the direct cost but return on investment possible.

2.8.0.2 Underestimation of Integration Challenges

- **Observation:** Finally, most of the research centered on the human factors and put a main emphasis on the positive effects that simply underestimate the difficulties and costs in integrating and managing human influence [123]. These could be, for instance, the costs incurred in the training of personnel or the difficulty of managing diversified teams.
- **Implication:** It also deals with these challenges and barriers that owe detailed study, and in this dimension of the cost-benefit paradigm, one can have a more balanced view about.

2.8.0.3 Overlooking Group Dynamics

- **Observation:** The literature, on the other hand, appears to focus singularly, in many cases leaving out group dynamics and interpersonal communication role when it comes to the impact brought about by single cognitive abilities and psychological traits on software development [122].
- **Implication:** The social factors impinging upon the efficiency, effectiveness, and subsequently upon the overall cost of software development, however, is perhaps something which has not received due attention. This is highly relevant—more than in any other context—since software development is basically a collaborative exercise.

2.8.0.4 Absence of Empirical Evidence

- **Observation:** One significant observation is that apparently there is an overt absence of empirical studies showing how well HF is integrated into the process of software development at the systemic level. Most of the current research in this domain is, by all means, mainly theoretical or qualitative in nature [124]. There actually is no actual and real, veritable, quantitative data that can prove and/or negate the cost-benefit implications of incorporating human factors into software development.
- **Implication:** Future research, with more robust empirical studies, would be able to yield substantial evidence that either supports or rejects the theoretical constructs being discussed in the current literature.

2.8.0.5 Methodological Limitations

- **Observation:** Another important limitation would be on the coverage of the existing research based on the methodological approaches incorporated. Most of the literature tends to cover small sample sizes, limited geographic areas, or be specific to industry or type with its software development [125]. These findings are, therefore, of low generalization value.

- **Implication:** This work highly, therefore, recommends future research to employ sound methodologies and high scope in order to make their findings widely applicable.

2.8.0.6 Neglect of Agile Methodologies

- **Observation:** Lastly, the literature to date has materially skewed to consider human factors in the traditional software development methodologies paradigm, such as the Waterfall Model.
- **Implication:** Most significantly, it appears to lack the human factor in a material way with modern agile methodologies [126]. This is a critical gap, given that these methods are rapidly becoming the norm in the industry.

In brief, this literature review is of immense value in understanding the human factor issues associated with software development, but seriously wanting in the cost-benefit dynamics understanding of its association with their integration. These gaps should be attended to in future research, in such a way that more comprehensive, balanced, and nuanced views on this important issue can be gained.

2.8.1 Identification of the gaps in the current body of knowledge

Looking through the exhaustive studies in the field of software development and its cognate fields, it comes as a pivot to realization that if there exists a gap of any note in this exhaustive literature, then that is it: very few exhaustive studies exist in the field of software development. "This way, it would be useful not just to draw out areas that further need to be probed but also to locate our research in the large framework of the set body of knowledge [131].

2.8.1.1 Sparse Quantitative Analysis on Economic Impact

From the literature, visible at first sight is the gap that relates to the deficient quantitative analysis regarding the human factor in software development. There have been studies conducted to explore human factors and their effect on the software development process [132]; however, further economic impact as a result of these is still a lesser-explored domain for a more pinpointed, accurate quantitative analysis. This area has, in fact, only seen a quantitative investigation of the cost-benefit analysis of human factors in the software development process sparingly [133].

Table 2.6. Table highlighting key limitations in the existing research on human factors in software development, emphasizing the need for more robust, empirical, and long-term studies.

Limitation Type	**Description**	**Impact on Field**
Empirical Evidence Shortage	Scarcity of studies with solid empirical data	Hinders the understanding of actual impacts and benefits
Quantitative Data Lack	Limited quantitative analysis on the subject	Affects the ability to make generalized and statistically sound conclusions
Long-Term Impact Studies	Focus mainly on immediate or short-term effects	Leaves uncertainty about the sustainability and enduring value of practices

2.8.1.2 Superficial Examination of Individual Traits

Further, while the human factors in the productivity and efficiency of software development teams are well-documented [134], most of these studies find that the depth of such research is shallow in the dissection of the role played by individual traits and how, collectively, they either help or hurt the process of software development. This resulted in such a study of cognitive abilities, personality traits, and emotional intelligence of software developers in isolation from each other, without a comprehensive work that would bind these issues to each other and examine their joint effect on the software development process [135].

2.8.1.3 Limited Focus on Psychological Traits and Decision-making

Still, much is to be desired in searching for the psychological traits of the software developer and his relevance to decision-making and judgment in the process of software development. While current literature does acknowledge the prevalence of cognitive biases in decision-making [136], this is a very confined study exploring psychological factors and their specific influence over the making of the decision in the context of software development.

2.8.1.4 Neglect of Integration Challenges

The literature does not cover the challenges that relate to integrating human factors within the software engineering process. All such benefits were usually described;

further research may be carried out to search for any possible obstacle, disadvantage, or resistance to its incorporation. For example, the training program may take much more time and resources on many challenges, such as the diversity of employees to be managed in a team and how to balance the productivity and the health state of the developers.

2.8.1.5 Under-researched Cost-Benefit Analysis

Last, concerning human factors, cost-benefit analysis in software development has received meager attention. The other studies reviewing cost-benefit analysis in software development discuss software tools and technologies, among other factors [138]. Clearly, here is an opportunity to cost-benefit analyze human factors in software development as a way to provide illuminating insights regarding how these factors can affect the economics of software development.

The discovery of these gaps in the existing literature accentuates the necessity for further research. In effect, these gaps call for a comprehensive cost-benefit analysis considering human factors in software development to throw light into the murky waters of economic factors and human traits interacting in software development.

2.8.2 The need for cost-benefit analysis of incorporating human factors in software development

Despite the growing attention to the human factor in software development, one particular aspect is still somehow neglected: the use of cost-benefit analysis for assessing the implications of incorporating human factors in the software development life cycle. With so much impact of human factors on the software development process, this becomes absolutely crucial to carry out the cost-benefit analysis in as detailed a manner as possible to make good and informed decisions.

2.8.2.1 Necessity for Cost-Benefit Analysis

1. **Quantification of Financial Implications:** The cost-benefit analysis indeed provides a systematic way of quantifying and comparing the pros and cons of a decision, policy, or investment. This analysis uses financial metrics of estimating the return on investment (ROI), and hence allows determining economic feasibility of the alternatives. Integrate with software development a cost-benefit analysis of human factors being recommended, considering the way in which economic implications of such a decision could be evaluated.

2. **Balancing Investment and Outcomes:** On the other hand, embedding human factors in the development of software usually involves investments, including funding for staff training and team building, as well as the adoption of new

procedures to cater to a variety of cognitive and psychological characteristics. The organization may also be under cost obligation to invest in advanced tools that will aid improved collaboration among their employees or the other stakeholders. Thus, while identifying the human factors may end in a better quality of software with the most efficient teams, the cost that comes with them needs to be accounted for.

3. **Understanding Long-term Benefits:** Besides, it is to be said that the impact of human factors on the software development process is not always recognized instantaneously from an objective perspective, but rather vice versa is likely to be occurred. Thus, some benefits carry along with them improved job satisfaction but are not transferred into immediate financial returns. They have a potential benefit that could reduce staff turnover, increase creativity, and boost the morale of the team. Conducting the cost-benefit analysis would therefore be a way of quantifying these long-term benefits and bring out a more encompassing view of the implications of the incorporation of human factors.

4. **Risk Mitigation:** Research has indicated that without human factors in the production, software has an increased risk of failure in terms of cost overruns, time delays, and poor quality. These are the costs of failing to include human factors and are clearly large, mandating some form of cost-benefit analysis in this context.

5. **Filling the Knowledge Gap:** Despite the apparent need, there is evidence lacking or no evidence to guide the cost-benefit analysis for incorporating human factors in software development. This knowledge gap could well be considered a barrier to the informed decision-making of the organization with respect to their practices in software development. It is therefore felt that further research in this area is needed to fill this gap in knowledge and provide the necessary tools to software development organizations, which could enable them to make informed decisions.

In general, human factors in software development come with many benefits but also have potential challenges. Conducting a cost-benefit analysis can offer an invaluable insight into the economic implication of this decision, therefore enabling organizations to optimize resource allocation with the goal of maximum ROI realization. It, therefore, becomes very obvious that there is a clear pressing need for the matter to be investigated further, more to the point of drawing out the cost-benefit analysis of incorporating human factors in software development, which is long overdue, requiring research as a topic.

Chapter 3

RESEARCH METHODOLOGY

3.1 Introduction

Cost-benefit of the human factors of embedding in software development takes into account a systematic method in which the methods that have to be applied are captured in a much more careful and rigorous approach. In this view, it follows that the following will depict the methodological approach towards conducting a systematic cost-benefit analysis of integrating human factors in the realm of software development. In one word, the approach is, sketchy—meaning just enough to point out the economic and quality effects of human-centric practices, such as software engineering, and to engage one with the key insights in a way that these practices can be optimized to enhance quality in software, develop efficiency, and dynamics. Here, the components for this section are presented.

- **Research Design:** This is the second section of why the study has a quantitative orientation and how this will help to provide support for a rigorous investigation to the research questions.
- **Instrument Development and Validation:** This section revises the design and refinement of the research instruments in capturing in detail the dynamics that are in existence. In particular, this section revises the design and refinement of the questionnaires for use by software developers and managers, for reliability and validity.
- **Data Collection Methodologies:** The strategies employed for data gathering are detailed, highlighting the diverse channels through which the questionnaires were disseminated and the criteria for participant selection.
- **Analytical Strategies:** An overview of the statistical techniques and tools deployed to analyze the collected data, aiming to extract meaningful patterns and correlations that inform the cost-benefit narrative.

To aid in the comprehension of this methodological journey, a schematic representation of the framework is proposed. Figure 3.1 encapsulates the stepwise process, from the inception of research design through to the analytical dissection of data, underscoring the integrated approach adopted in this study.

3.2 Research Design

Since the holistic perspective demands a critical role to be played in regard to the effectiveness of adopting human factors in software engineering practice, the research design makes and enforces clear of the choice over the maze of quantifying the concrete and mostly enigmatic elements. This section clearly elucidates the structured approach taken in regard to the chosen quantitative research design through detailing the data collection and specifying the methods of analysis used.

3.2.1 Design Choice:

This research was built on a quantitative research design in a systematic manner. The selection of this design seemed quite apparent since the whole effort needed data that could be quantified and statistically measured, preferably providing practical acumen related to commercial and performance-based situations following the integration of human factors with the technology of software engineering. The usage of quantitative research seeks to provide the preciseness and objectivity required to establish the relationships, differences between variables, or to generalize results based on the sample studied for setting within the software development industry.

3.2.2 Data Collection:

At the core of this quantitative approach were structured questionnaires which were carefully designed by the authors to elicit numerical data reflective of that which could explain the perceived and actual costs and benefits in human factors integration. For example, they made it possible to solicit quantitative data from the respondents concerning investments in training, metrics of productivity, quality indices of software, indicators of well-being of employees. These conditions yielded uniform, consistent data from a diversified participant pool, which is representative of software developers and managers with direct experience in, or oversight of, human factors integration.

3.2.3 Analysis Method:

Above all, this research is articulated based on empirical analysis that has been afforded a backbone by the application of a suite of statistical tools and techniques, which can be able to dissect the data that has been collected and interpret it. Where applicable, this study made use of SPSS (Statistical Package for Social Sciences) for its very strong ability aspects of data management and analysis—via the even availment of operations such as correlation analysis, regression analysis, and variance analysis. More or less, Microsoft Excel complemented the process of analyzing data and visualizing data toward the preliminary analysis, which was to further communicate with the results of statistics.

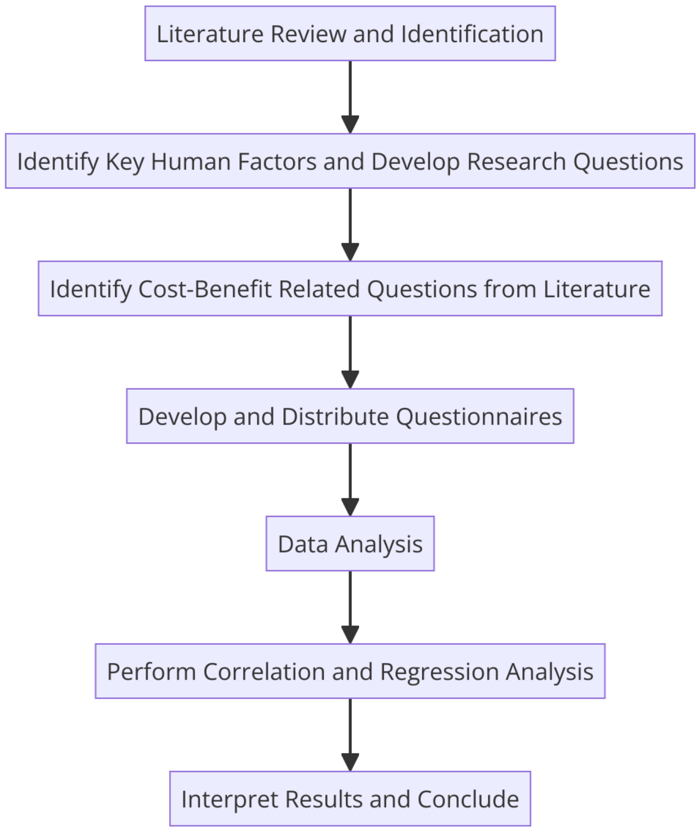

Literature Review and Identification
Identify Key Human Factors and Develop Research Questions
Identify Cost-Benefit Related Questions from Literature
Develop and Distribute Questionnaires
Data Analysis
Perform Correlation and Regression Analysis
Interpret Results and Conclude

Figure 3.1: Flowchart Illustrating the Research Methodology for Evaluating the Impact of Human Factors and Cost-Benefit Analysis in Software Development

3.3 Instrument Development

3.3.1 Development Process

The core of this study was at two systematically designed questionnaires with thorough examination of data gathered from literature review;

- **Software developers** who are experiencing the factors at first hand are responsible for the development of software
- **Managers** within the software development industry who are directly involved in the matters of investment which was important to undertake the cost and benefits.

Both of these questionnaires were developed after a thorough examination of the literature and studies.

Keeping in mind the existing gap in literature and the lack of first-hand data regarding the costs and benefits of incorporating human factors in software development, these questionnaires were meticulously designed and reviewed and repeatedly revised to bridge the existing gap in literature. They aimed at not only getting the developer's and manager's perception about the importance and impact of most recurrently occurring human factors in literature—such as adaptability, communication skills, problem-solving abilities, teamwork, and continuous learning—but to also to study the economic implications of the human factors.

The development of the questionnaire was a detailed process which was mostly developed after a detailed analysis of the literature relevant to our topic. The main aim was to develop an effective instrument to capture the impacts of human factors and analyse their cost and benefits from the viewpoint of both developer and managers.

3.3.1.1 Step 1: Literature Review

The initial step involved a thorough examination of the existing literature and scholarly review. The main objective of this literature review was;

- Identify significant human factors impacting software development.
- Understand the roles and implications of these factors.
- Determine potential indicators for both costs and benefits.

3.3.1.2 Step 2: Identification of Key Factors

From the literature review, five key human factors were identified as critical to software development success:

- Adaptability
- Communication Skills
- Learning and Development

- Problem-solving Skills
- Teamwork

These steps provided the basis of the initial questions in both of the questionnaires for both managers and the developers.

3.3.1.3 Step 3: Formulation of Questions

The initial questions were developed with the help of the most commonly occurring human factors in software development. The impact of all five most common human factors were asked on a Likert scale to find out the most impactful human factor and their correlation with one another. The questions were designed to be quantitative to make a better statistical analysis.

Table 3.1. Literature-Derived Human Factors and Corresponding Questionnaire Items

Human Factor	**Corresponding Questionnaire Items**
Adaptability	"Rate the impact of adaptability on software development quality."
Communication Skills	"Rate the impact of communication skills on software development quality."
Learning & Development	"How frequently do you receive training focused on enhancing learning and development?"
Problem-solving Skills	"Rate the impact of problem-solving skills on software development quality."
Teamwork	"Rate the impact of teamwork on software development quality."

3.3.1.4 Step 4: Incorporating Cost-Benefit Analysis

Given our main focus on analysing the impact of human factors while keeping in mind an economical impact of the human factor, questions related to costs and benefits were also included in the questionnaire. This step included integrating questions related to costs and benefits into the questionnaire:

- **Cost-Related Questions:** Focused on training investments and time resources.
- **Benefit-Related Questions:** Aimed at measuring outcomes such as defect reduction, improved delivery timelines, and job satisfaction.

It should be kept in mind that due certain limitations and lack of data, we were unable to get numbers in terms of actual money spent on the human factor incorporation. The main method was to find factors from literature that could be translated in terms of costs and benefits. The questions aimed at the perceived costs included

- training investments
- time resources

and the ones related to benefits included

- reduction in defects
- improved delivery timelines
- increased job satisfaction

Questions related to the costs and benefits of incorporating human factors into software development were added to both the questionnaires developed for managers and developers in order to get a better understanding of the viewpoints of both stakeholders.

Table 3.2. Cost-Benefit Related Questions Derived from Literature

Aspect	**Questionnaire Items**
Training Investments	"On average, how much financial investment is directed towards training focused on human factors?"
Time and Resources	"How would you rate the time and resource cost associated with enhancing human factors?"
Defect Reduction	"Since focusing on human factors, have you observed a reduction in defects or errors in your work?"
Delivery Timelines	"How has emphasizing human factors impacted your speed of delivery?"
Job Satisfaction	"How has the focus on human factors affected your overall job satisfaction?"

3.3.2 Questionnaire Content

Both questionnaires commenced with screening question about the role of the respondents to ensure they are relevant to the study and to make sure we get appropriate data. After the screening question, both the questionnaires included demographic data to find out the respondents age, genders, experience and educational background. The main aim of this approach is to find out the impact of these demographics on the users and to see the difference in opinion based on the user demographics. It should be noted that large portions of both questionnaires contained questions regarding the measures that can be translated to tangible outcomes and regarded as the measure of costs or benefits such as changes in defect rates, project delivery speeds, and overall job satisfaction.

3.3.3 Justification for Questionnaire Structure

The questionnaire was developed and drafted to the point where it is enough in all aspects when analyzing facets or influences that human factors encompass with software development. The section avails the reason behind the design of the questionnaire and the sections.

- **Purposeful Segmentation:** The questionnaire was segmented into distinct sections to capture a holistic view of human factors in software development. This segmentation enables:
  - A detailed examination of each identified human factor.
  - The assessment of their individual and collective impacts on software quality and development processes.
- **Inclusion of Demographic Section:** Demographic data were solicited to:
  - Understand the diverse backgrounds and experiences of respondents.
  - Analyze how these variables may influence perceptions of human factors in software development.
  - Provide insights into how demographics correlate with the perception of costs and benefits related to human factors.
- **Employment of Likert Scale:**
  - The use of a Likert scale facilitates quantification of perceptions regarding the significance and impact of each human factor.
  - It allows for the collection of nuanced data on respondents' attitudes, providing a basis for statistical analysis to identify patterns or correlations.
- **Systematic Analysis Potential:** The structured approach to the questionnaire design is intended to:
  - Facilitate a comprehensive analysis of the collected data.

    - Enable the examination of the intricate relationships between human factors and their impacts on software development outcomes.
    - Support the identification of significant trends and insights that can inform future strategies and practices in software development.

- **Rationale for Sections:**
    - **Demographics:** To ensure the representativeness of the sample and understand the influence of background factors.
    - **Human Factors Impact:** Dedicated sections for each key human factor to delve into its specific impact on software development.
    - **Cost-Benefit Analysis:** To evaluate the economic implications of integrating human factors, aligning with the study's objectives.

- **Contribution to Research Objectives:** The deliberate design of the questionnaire is closely aligned with the research objectives, facilitating:
    - The identification of pivotal human factors in software development as perceived by different stakeholders.
    - An exploration of the cost-benefit dynamics associated with these factors.
    - An enriched understanding of how demographics influence perceptions and experiences related to human factors in software development.

This structuration approach is going to make sure that the data provided by the questionnaire is not only comprehensive in its approach but covers itself to be in harmony with the stated grand objectives of this particular research, and consequently, make a well-rounded exploration related to human factors in software development from various perspectives.

3.3.4 Question Development and Refinement

Questionnaire development was done most carefully and in detail, particularly through successive improvements directed at enhancing clarity, objectivity, and relevance of questionnaires towards the goals of the research survey study. The following section lays emphasis on each stage of the refinement of questionnaires based on expert panels and criteria involved in the judiciousness of the concern in improving the methods.

- **Iterative Refinement Process:**
    - **Initial Drafting:** Based on the literature review and identified key factors, initial question sets were drafted focusing on human factors and their cost-benefit implications in software development.

Table 3.3. Structure and Content of the Developer Questionnaire

Section	**Focus Area**	**Description**
Demographics	Participant Background	Age, gender, years of experience, highest educational qualification, current role, organization size, and type.
Impact of Human Factors	Perceived Impact on Software Quality	Participants rated the impact of specific human factors (e.g., adaptability, communication skills) on software quality on a Likert scale from 1 (No Impact) to 5 (Significant Impact).
Frequency of Training	Training on Human Factors	Questions regarding how often participants receive training aimed at improving human factors like problem-solving and teamwork.
Organizational Support	Support for Human Factors Enhancement	Assessment of the degree to which organizations support the enhancement of human factors and its influence on satisfaction and productivity.
Tangible Outcomes	Outcomes of Focusing on Human Factors	Questions aimed at understanding the tangible effects of emphasizing human factors, such as changes in defect rates and project delivery speeds.

Table 3.4. Structure and Content of the Manager Questionnaire

Section	**Focus Area**	**Description**
Demographics	Participant Background	Questions about age, gender, years of experience, highest educational qualification, number of developers managed, organization size, and type.
Perceived Impact	Impact on Software Quality	Managers rated the impact of human factors (e.g., adaptability, communication skills) on software quality on a Likert scale from 1 (No Impact) to 5 (Significant Impact).
Financial Investment	Investment in Human Factors	Insights into the financial investment and resource allocation towards enhancing human factors in software development.
Observational Insights	Changes Post Human Factors Initiatives	Questions aimed at gathering observations on changes in software quality, team dynamics, and project delivery timelines after initiatives focused on human factors.

  - **Expert Feedback:** Drafts were presented to a panel comprising software development professionals and academic researchers.
  - **Revisions:** Based on feedback, questions were refined for clarity, relevance, and neutrality to eliminate any bias and ensure comprehensiveness in covering the study's scope.

Table 3.5. Summary of Feedback and Revisions

Draft Round	Feedback Summary	Revisions Made
1	Clarify certain human factor definitions.	Definitions refined and examples provided.
2	Suggestions to add questions on specific cost elements.	Added questions on training costs, time investments.
3	Recommendations to reword questions for better respondent understanding.	Simplified language and restructured sentences.

- **Criteria for Question Effectiveness:**
  - **Clarity:** Questions must be easily understandable without ambiguity.
  - **Relevance:** Each question must directly contribute to exploring the research objectives.
  - **Neutrality:** Questions should be framed to avoid leading responses or implying preferred answers.
  - **Measurability:** Responses should be quantifiable, particularly through the use of Likert scales for subjective assessments.
- Validation for Reliability and Validity:
  - A subset of the developed questionnaire was pilot-tested with a smaller, representative group to assess understanding and response consistency.
  - Adjustments were made based on pilot feedback to further enhance the questionnaires' reliability and validity.

The thorough process of iterative development and refinement can, therefore, be seen to carry the final questionnaires through to be somewhat regarded as robust and effective tools in acquiring meaningful data, well-designed to capture the nuanced impacts of human factors in software development; thus yielding insights that are pivotal in the costing versus benefits analyses at the core of this research.

3.4 Data Collection

This sub-unit gives the application strategy used during the data collection, which was important in aiding the analysis of the cost-benefit implication of integrating human factors in the processes of software development. It will further include the use of Google Forms in receiving the questionnaires, and it was necessary, and some of the ways used in getting to the potential respondents.

- **Choice of Google Forms:**
  - **User Accessibility:** Google Forms is widely accessible. It requires no special software or subscription. Therefore, making it an inclusive option for participants across diverse geographical locations.
  - **Ease of Use:** Its intuitive design and straightforward interface ensured that respondents could complete the questionnaires without difficulties, thus improving the quality of data collected.
  - **Data Organization:** Automatic data organization into spreadsheets facilitated preliminary analyses and ensured data integrity.
- **Distribution Channels:**
  - **Professional Networks:** Utilized LinkedIn to reach a broad range of professionals in the software development industry. Specific groups and forums related to software engineering and project management were targeted.
  - **Social Media Platforms:** Shared through X and Facebook groups focusing on technology and software development to capture a diverse respondent pool.
  - **Personal and Peer Networks:** Then the survey link was bounced to be forwarded through WhatsApp in industry contact contacts in their personal and professional networks. It was urged upon with industry contacts to forward the survey link to respective their peers who are relevant. This research methodology also pares in that the research has been able to crack through professional tight-knit communities or participants otherwise not covered on any other social media platform.
  - **Direct Outreach:** Beyond group shares, individual messages were sent to known professionals within the industry. This personalized approach ensured the participation of individuals with specific insights into the integration of human factors in software development.
- **Sampling Techniques and Screening Questions:**

Table 3.6. Summary of Data Collection Channels and Responses

Channel	Responses (Developers)	Responses (Managers)
LinkedIn	150	60
Twitter	80	20
WhatsApp	100	30
Direct Outreach	18	14

  - **Purposeful Sampling:** Aimed to include individuals directly involved in or knowledgeable about software development, enhancing the relevance and quality of the data.
  - **Screening Process:** Embedded a screening questions at the beginning of the questionnaire. This was to ensure participants had relevant experience or roles in software development.

- **Data Collection Timeline:**
  - **Start and End Dates:** Data collection was initiated on November 13th, 2023 and concluded on November 27th, 2023, allowing ample time for a wide range of participants to respond.
  - **Response Monitoring:** Regular checks were conducted to monitor the influx of responses and ensure a diverse sample was being reached.

The process of data collection, however, used the tools and channels selectively to reflect on strategic sampling techniques that were intended to ensure criteria for high quality, pertinent data are established. The methodologies used in this ensuring the considerations come out are exhaustive perspectives both from the developers and managers point of view on inclusion of human factors in general and the economic implications during software development.

3.5 Data Analysis

The responses derived through the questionnaires have shown that quantitative data is treated with a lot of care and abuse. One tool and application used for analyzing quantitative data derived from the questionnaires in great detail is the use of the all-powerful Microsoft Excel, which is very useful for conducting regular statistics. So, the journey of analysis started through an application of descriptive statistics, in fact, which is the building of a basis of a data set through techniques of means, medians, standard deviations, and patterns of distribution.

3.5.1 Analysis Stages:

- **Descriptive Statistics:** This summarized the data to find out the average tendency and variability that apply. This provided insights into trends, for example, the average impact of human aspects over software development quality, standard deviation in responses concerning cost perceptions, benefit perceptions.
- **Comparative Analysis:** Next to an overview of findings in the forms of descriptive analyses, there were comparative analyses done in order to outline differences and similarities within data subsets. It was about comparisons within different subsets of responses, like, for example, between different types of developers or between different demographic data groups of participants in the survey.
- **Correlation Analysis:** A correlation analysis was done to find out relationships between human factors and their perceived cost-benefit implications in software development. It strictly identified the strength and direction of associations between the relationships indicated above.
- **Regression Analysis:** The final phase of data analysis comprised regression analysis to predict the influence of respective human factors on positive outcomes of software development. In this step, micro probing, within the specific impacts of human factors, was reported for adaptation by the adaptability and communication on software quality, time saving in software development and cost savings, respectively. The workflow from descriptive statistics through regression analysis is fully elaborated in the figure 3.3 below to structure a response and thus indicate clearly the progressive nature of analytical techniques applied to the dataset under consideration:

3.5.2 Visualization and Reporting

In the processes, in order to make findings clear and compelling; in that way, it makes complex statistical relationships and trends which are presented succinctly, appealing in the interpretability of data. Statistics for further analysis can then be visualized by use of Excel charting/graphing tools in subsequent presentations.

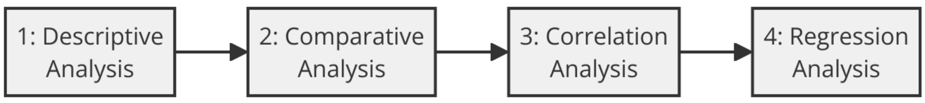


Figure 3.3: Data Analysis Workflow

Describing each step conducting analysis and naming the specific software, this modified part informs how the data were analyzed in a structured way making the research findings made transparent and subjected to methods of conducting reproducibility.

3.6 Ethical Considerations

The ethical soul of this study is to ensure that it stands by the policy and standards already established. Effecting the following measures sustains the principles of confidentiality, anonymity, and respect for participants in effecting these measures: ethical sensitivity of this study.

- **Confidentiality and Anonymity:** No personal identifiers such as names or specific organization details were collected at any stage of the questionnaire.
- **Non-Discrimination and Respect:** The questions were designed to be culturally sensitive, containing no questions considered discriminating against or fostering disrespect toward any religious, cultural, or ethnic grouping. In the stemming down of the questions to the participant, their formulation was such that it was all-embrasive and permissive of the different views to be presented.
- **Risk Minimization:** Assessment was done to eliminate any probable risks to respondents, it was noted that the research would get involved with very low risk, minimal risk intensity, since it only concerns professional opinions and related experiences rather than personal or sensitive information.

The observation of these ethical postulations was closely succeeded by the rigorously observed procedure to safeguard the participants welfare and reputation thereby be responsible and respectful in the manner of engagement in the research. The measures adopted in these respects underline the seriousness afforded to this issue in ethical and legitimized research work and add to the dependability and credibility of the findings in the study.

3.7 Limitations

This study's design and execution, while meticulously planned, inherently face several limitations that could influence the findings' applicability and interpretation. These limitations include:

- **Perception-Based Data:** Respondant's perceptions and recalling probably would make the reflection of the behaviors and the outcomes with reality, which are in the case of the self-reports. Therefore, the study reacts accordingly over this direction.

- **Lack of Numerical Data:** The major challenge is by the fact that there lacks concrete numerical aspects on the financials of how the human factors will be incorporated into the software development hence limiting ability to make an actual quantitative assessment of the real costs and benefits.
- **Sample Representativeness:** This was a research purposive sampling technique applied in this research; as a result, it may not represent the other parts of the software development industry. Because of these conditions and settings, it may bring about problems with generalization of the findings.
- **Potential Response Bias:** This is important since due to the quality of the data being question-based, this leads participants to be exposed to succumb to response bias, where, for example, they are to answer what one sees as the social norm that is expected or desirable rather than their real views.

It should also be noted, however, that these have to be taken into due consideration when interpreting the results of the study. It suggests exactly where and how the methodology applied in future research will have to be revised in the light of the present study and must hold special significance on the path of evaluation of cost versus benefit in including human factors in the process of software development. Efforts should, hence, carry on to handle these limitations so that the field can itself rack up a more structured and holistic understanding regarding the contributions human factors make up to improving the processes and products of software development.

3.8 Conclusion

The entire proposed methodological framework deployed toward the complexity in integrating human factors in the software development lifecycle through a cost-benefit analytical eye was described in this chapter. In this research, structured research design was adopted, which is thoroughgoing and powerful in collecting pivotal numerical data, through intricately designed questionnaires both for developers and managers. This strategic approach was helpful not just for the distillation of empirical evidence but also laid down the grounds for statistical analysis with precision, which unfolds the economic implications also interwoven with the integration of human factors in software development.
Key highlights of our methodological journey include:

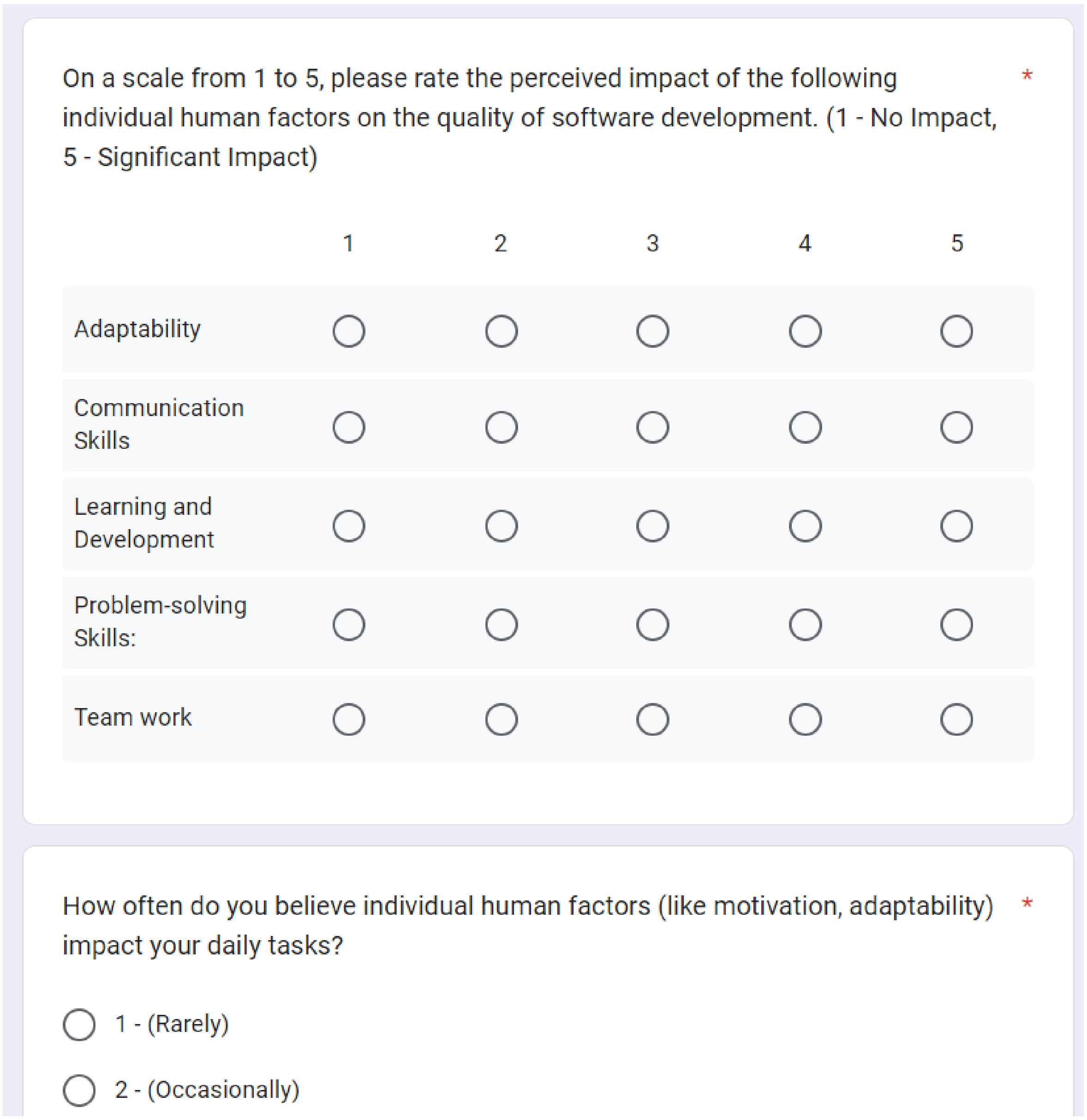
- **Innovative Instrument Design:** The formulation and validation of questionnaires were of very much importance, representing perceptions of developers and managers in detail about the role and effects of human factors in the context of software development. It is expected that this instrumental design will open the ways to other research methodologies.

- **Strategic Data Collection:** An adoption of a hybrid approach was by the use of different channels to make sure there was a strong collection of data; this way, diversity and reliability of the sample became better. The selection of such a research methodology points to a industry snapshot.

- **Analytical Rigor:** The nature of the present study, therefore, places great emphasis on empiricism in the use of refined statistical tools in the processing of data. The approach to the study through descriptive and inferential statistics makes this study not only a quantification of the real scenario but the analysis of the complicated paradigm in between humanly factors and their connotations.

The insights from all these methodological excursions are bound to add significantly to informed decision-making in the area of creating software. In this respect, the current study adds depth to academic discourses insofar as it articulates the economic impacts of integrating human factors and provides orientation on how better to optimise development processes. This, therefore, implies that reflections drawn from the study findings should act as a clarion call for strategic realignment of span of resources and practices that must accommodate the multifaceted benefits that the integration of human factors brings to the fore.

The insights from all these methodological excursions are bound to add significantly to informed decision-making in the area of creating software. In this respect, the current study adds depth to academic discourses insofar as it articulates the economic impacts of integrating human factors and provides orientation on how better to optimise development processes. This, therefore, implies that reflections drawn from the study findings should act as a clarion call for strategic realignment of span of resources and practices that must accommodate the multifaceted benefits that the integration of human factors brings to the fore.



On a scale from 1 to 5, please rate the perceived impact of the following individual human factors on the quality of software development. (1 - No Impact, 5 - Significant Impact) *

	1	2	3	4	5
Adaptability	○	○	○	○	○
Communication Skills	○	○	○	○	○
Learning and Development	○	○	○	○	○
Problem-solving Skills:	○	○	○	○	○
Team work	○	○	○	○	○

How often do you believe individual human factors (like motivation, adaptability) impact your daily tasks? *

- ○ 1 - (Rarely)
- ○ 2 - (Occasionally)

Figure 3.2: Screenshot of Questionnaire Layout on Google Forms

Chapter 4

DATA ANALYSIS

4.1 Brief Overview

The next chapter discusses the data analysis of the consciously prepared questionnaires that were targeted at two most important groups in the software development domain; the developers and managers. Although belonged to different roles and interests, the common objective is of the quality of the software product. Altogether, the quantitative approach will mostly rely on numerical ratings that participants give to reflect their underlying perception and experience of key human factors: adaptability, learning and opportunity in communication skills and development, and other vectors—such as those that determine the frequency and influence of observed factors on software quality and team dynamics.

The findings for this kind of multidimensional data collected were deciphered using help from certain cloud-based software and statistical tools. Microsoft Excel helped at the preliminary stages of data sorting, cleaning, and even some initial analysis, besides which it seems to most fittingly say that it is an elegant tool with the widest feature set and is familiar to many in the first place. Hereafter, preliminary data analysis and sifting through data for correlation and regression were also done using Excel. This means it is this layered approach that allowed the allowance not only of descriptive data but enabled the inferential statistics with the discovery of patterns, trends, and significant insights linked to the research purpose.

Our form of analysis has mainly been about attempting to explore and compare the view of developers with managers on different human aspects and what impact they had on the software development process. In this regard, efforts were made to explore the role and effectiveness pertaining to elements like communication skills, adaptability, learning opportunities, and organizational support, causing the success of the project, the good quality software delivery, and the maintaining of a proper balance of team dynamics. We then compared the generated insights from both categories of respondents to establish commonalities and divergences of the responses. Such results are useful in the devising of measures that would increase cooperation and productivity between software development participants towards leading to job satisfaction and the crafting of quality software products.

This paper, therefore, gives profound analyses fundamental for the achievement of the set research objectives herein; to understand the roles that identified human factors play in software engineering; evaluate the effectiveness of the current state-of-the-practice to address the these factors; and subsequently, derive recommendations based on evidence for improved process and outcome. The value of such analysis made will further expand the knowledge corpus of software engineering and project management in such a manner that assures better establishment and maintenance of more resilient, productive, and human-centric software development environments.

4.2 Data Cleaning and Preparation

4.2.1 Data Cleaning

Cleaning up data is not just important; it laid the bedrock of our analysis. It ensured that the findings came out as accurate and reliable as possible; this included organizing responses collected through Google Forms and had inputs from, among other things, sorting data through Google Sheets and Excel to arrange and group responses and aid in the creation of frequency tables.

- **Software Developers:** A total of 348 responses were collected, offering diverse insights into the development process.
- **Managers:** Received 124 responses, shedding light on managerial perspectives regarding the integration of human factors.

4.2.1.1 Addressing Missing Values:

Since the dataset was set up with a compulsion 1-5 Likert scale and all the responses set as compulsory, we cannot encounter missing data in the first place. This was, however, scrutinized through careful inspection, taken with great care, not to make our analysis get compromised.

4.2.1.2 Outlier Identification:

An outlier, particularly in Likert scale data, can spoil the results of the analysis altogether. We thus set out in our review to find any out of the ordinary response patterns, like uniformly responsive items suggestive of inattentive participation. No important outliers were revealed, and the data set may be considered reliable.

4.2.1.3 Accuracy Verification:

To ensure accuracy, cross-verification of inputs in Google Forms and compiling Excel were done manually, obviously involving standardizing full-length questions into some short and exact phrases, improving their clarity and making it easier for further analysis.

4.2.2 Data Transformation

The main change effected was the standardization of column names in order to easily reference them while analyzing. A very long question, "How would you rate your ability to adapt to changes in project requirements," for example, was renamed to read "Impact Adaptability."

4.2.2.1 Numerical Coding:

Systematic measurements for all the responses in forms of questionnaires were coded numerically between 1 to 5 for quantitative purposes. Those encodings permitted us to answer wound statistical methods, evaluate the general response profile within the data as well as variabilities besides conducting such cues as tests correlation and regression test.

4.2.2.2 Preparation Outcome:

The success of our preparation dataverse undoubtedly was a clear-cut effort in such a way that the proper analysis could run through. It would be clear through depicting every data in a visual way and cleaning and transforming them so they could be fit for the analytics tools, allowing the proper grounds to make the right findings regarding the feedback given by the respondents.

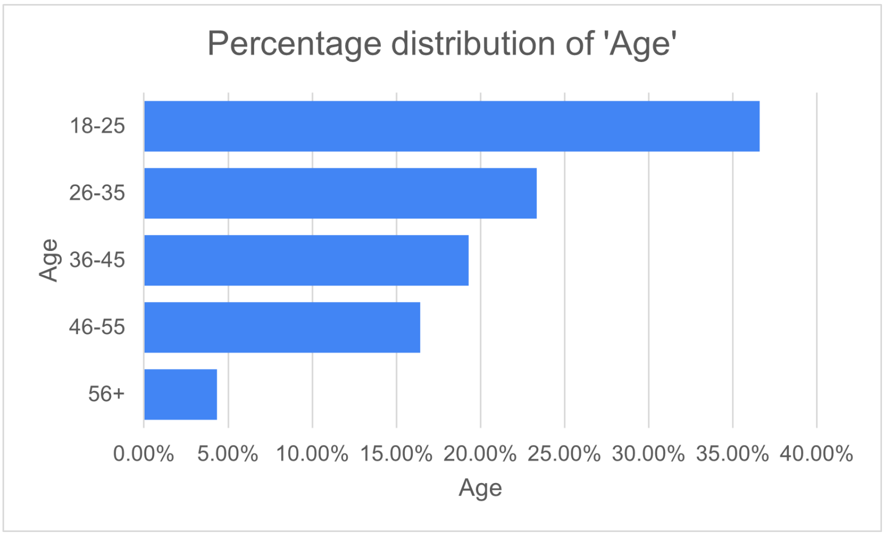
4.3 Descriptive Statistics

4.3.1 Participant Demographics

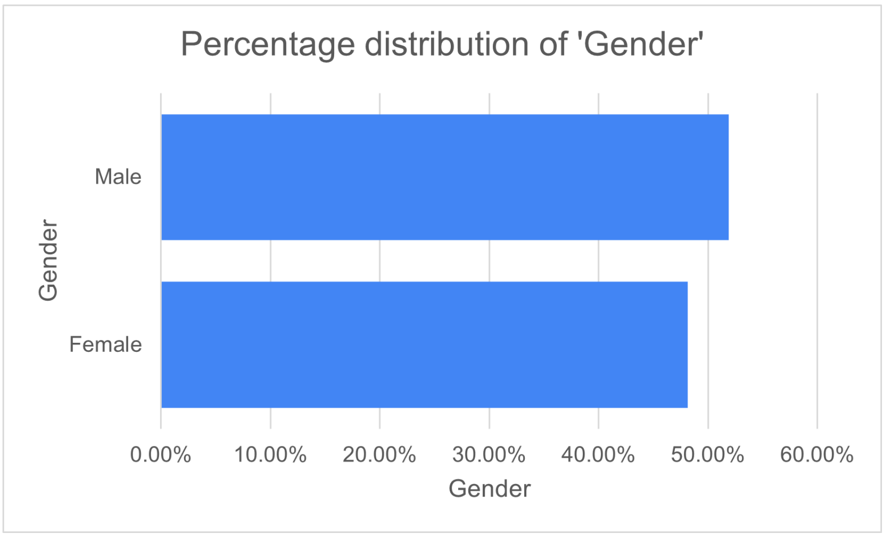
Divisions of people's survey questionnaires were put into two. Those whose responsibilities and interest in the software development business are staged. The first category includes the people who are the developers and the managers of the software. Below are the demographics of all our respondents. It summarizes the demographic details which epitomize each of the groups. The tables ahead are the tables of the most frequently occurring categories which are in the demographic data.

4.3.1.1 Developers:

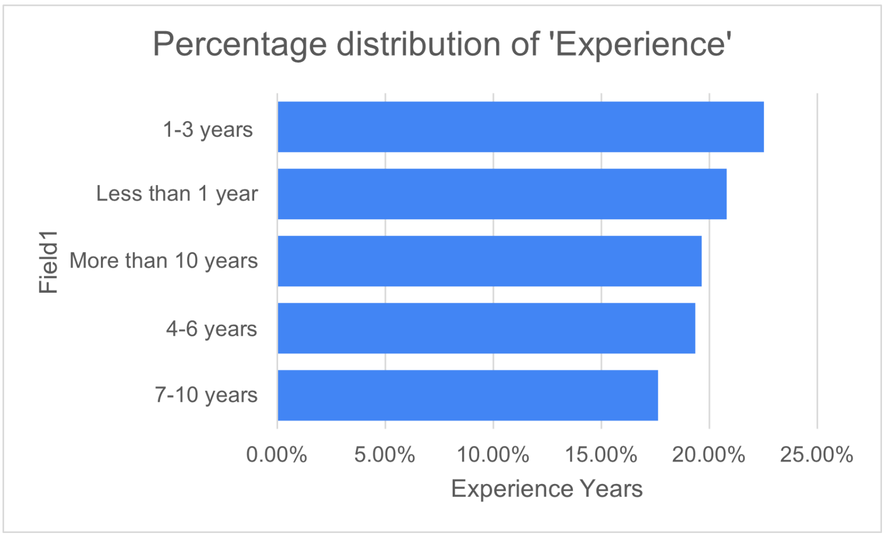
1. **Age Distribution:** There was a broad age range for developers. They had a significant concentration in the younger age brackets. This indicated a vibrant workforce predominantly in the early stages of their career.



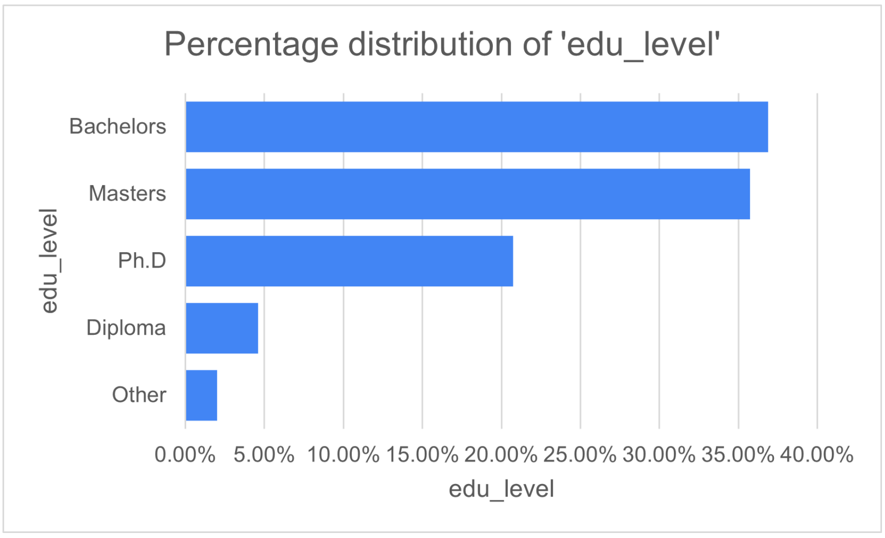
2. **Gender Composition:** Our analysis shows that a male-dominant demographic is within the developer group. This aligns with broader industry trends.



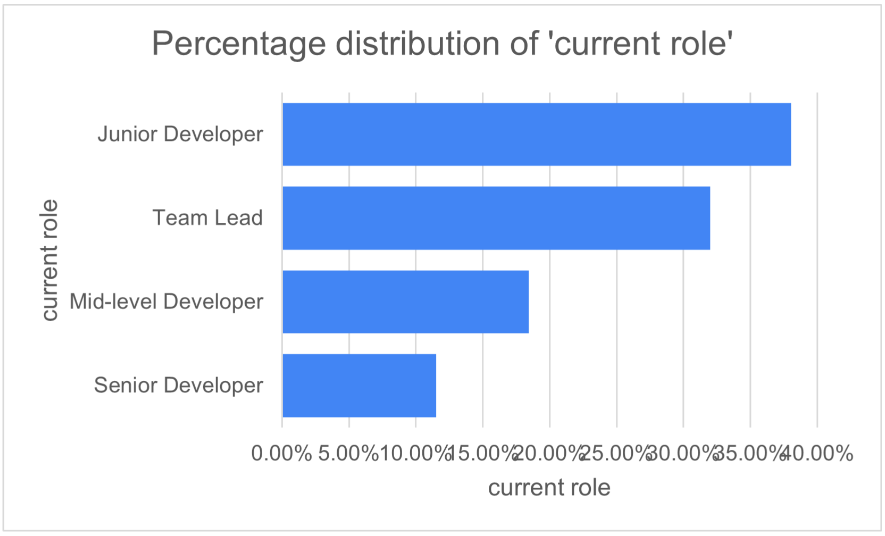
3. **Experience:** The developers' experience levels varied, with a notable presence in the 1-3 years range, suggesting a mix of burgeoning talent and seasoned professionals.



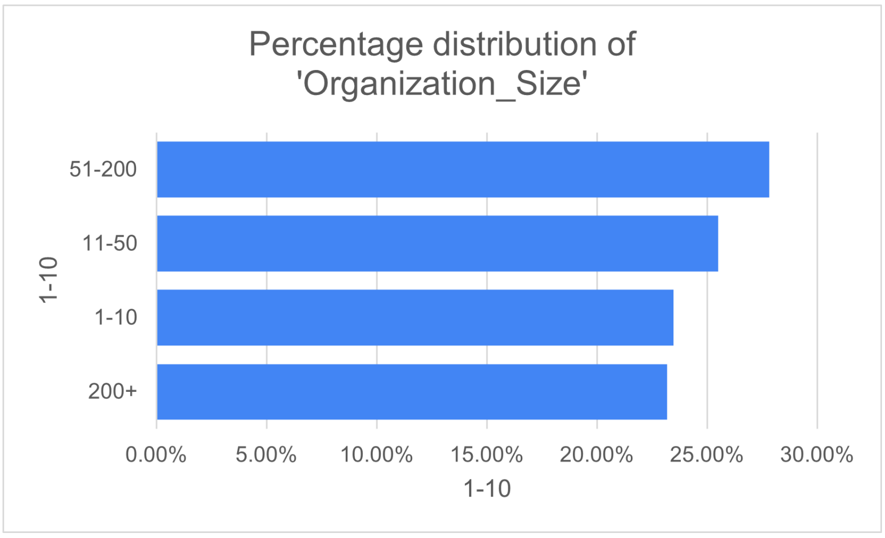
4. **Educational Attainment:** A bachelor's degree emerged as the most common educational level, highlighting the foundational role of undergraduate studies in software development careers.



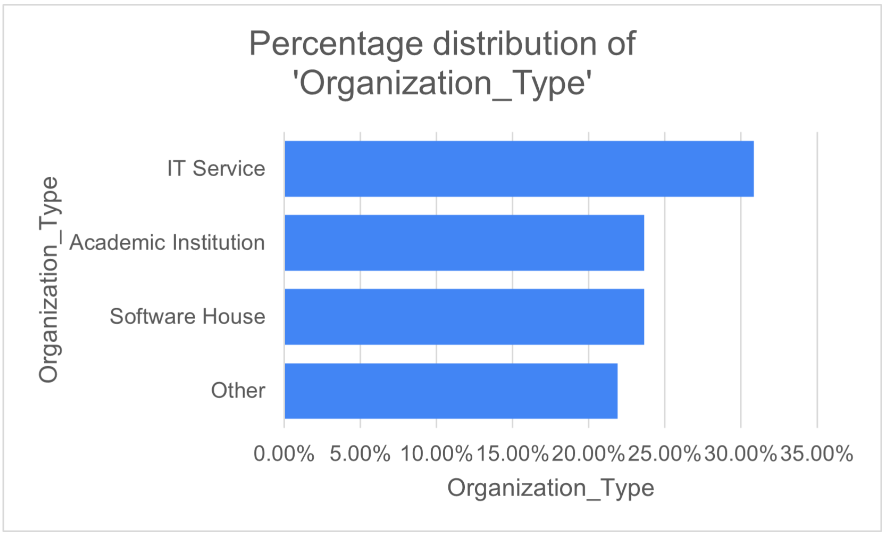
5. **Role within Organization:** Junior Developer roles were prevalent, suggesting that our respondent base mainly consisted of professionals at the onset of their software development journey.



6. **Organization Size:** The majority of developers worked in mid-sized organizations, indicating a dynamic work environment with ample opportunities for growth and collaboration.

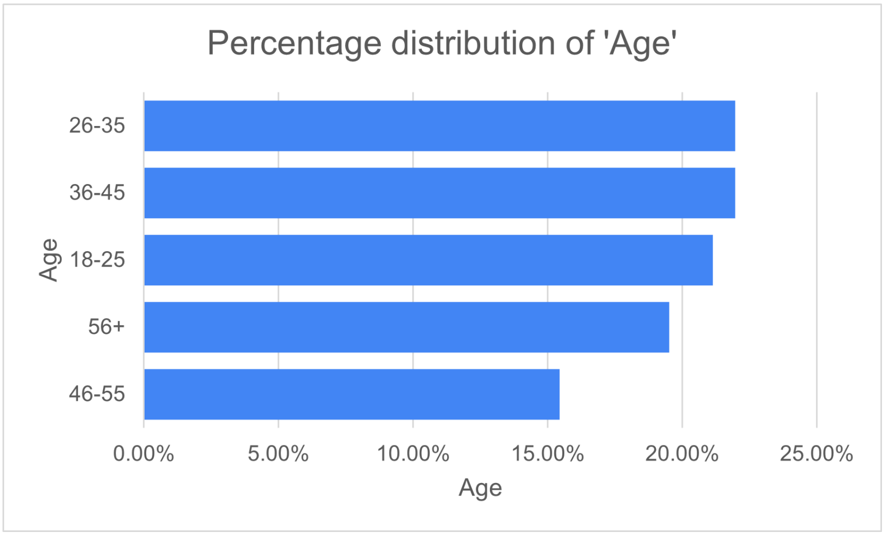


7. **Organization Type:** IT-service firms were the predominant employer type, underlining the sector's crucial role in providing software development services.

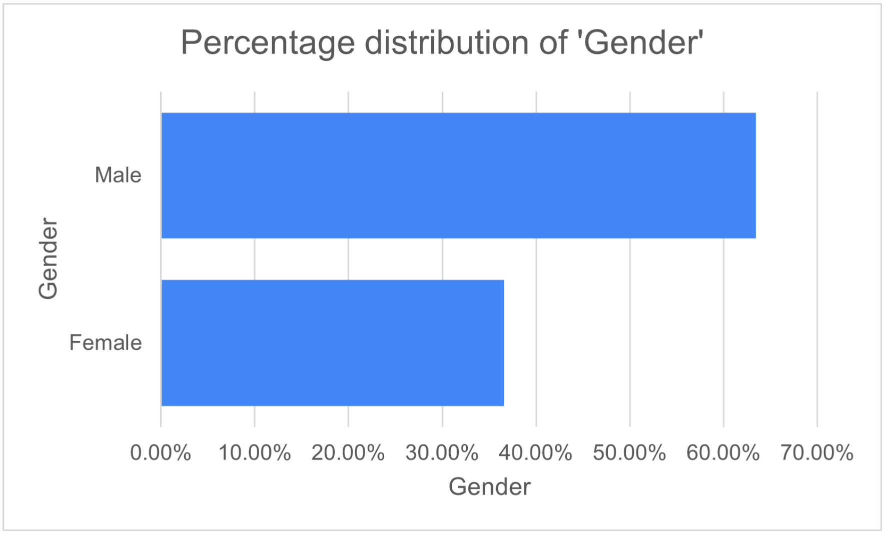


4.3.1.2 Managers:

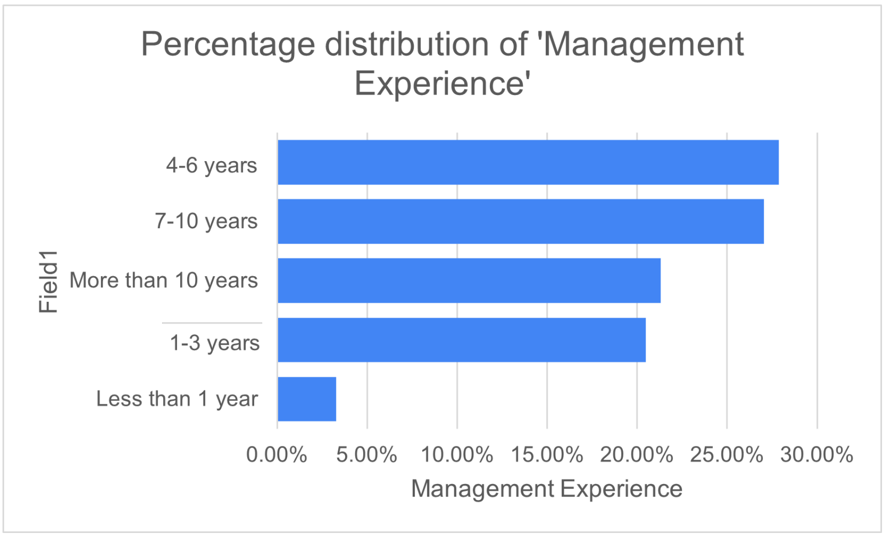
1. **Age Distribution:** Manager respondents tended to be older, with a concentration in the 36-45 age range, reflecting the typical career progression into managerial roles.



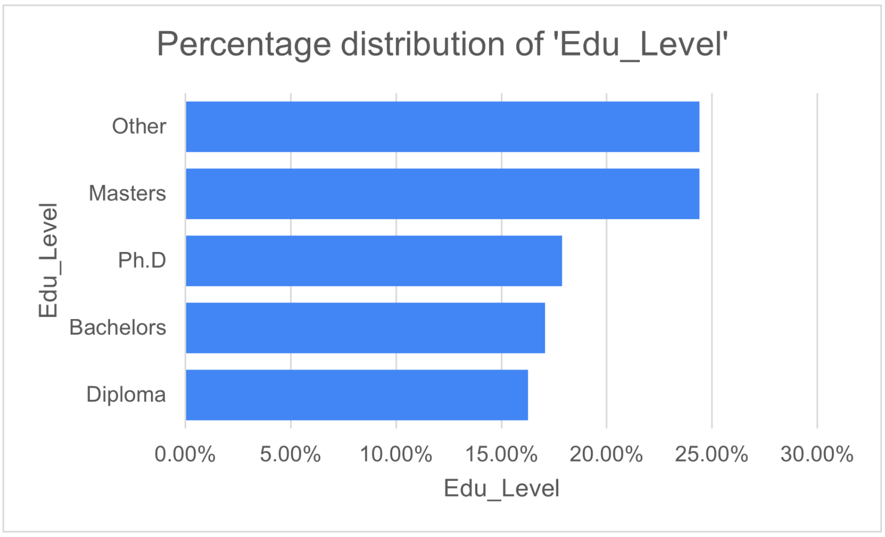
2. **Gender Composition:** Similar to the developers, the manager group was also male-dominated, mirroring the gender disparity observed across the tech industry.



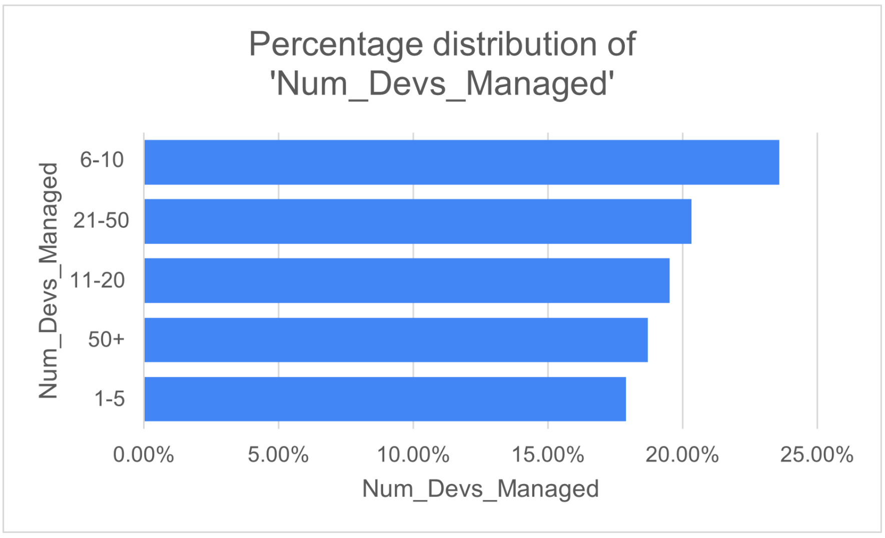
3. **Management Experience:** A significant number of managers reported 4-6 years of management experience, indicating a transition phase from hands-on development to leadership roles.



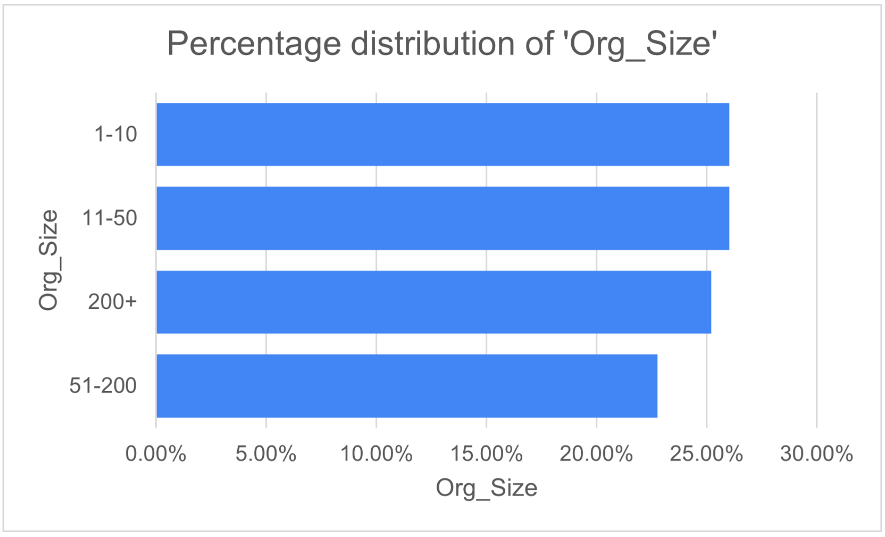
4. **Educational Attainment:** Masters-level education was more common among managers, suggesting that higher education may play a role in climbing the corporate ladder.



5. **Number of Developers Managed:** The data shows a wide range in the number of developers managed, with a focus on smaller teams, possibly allowing for more direct and effective team management.



6. **Organization Size:** Managers often worked in smaller organizations compared to developers, which might influence management styles and project approaches.



7. **Organization Type:** Similar to developers, IT-service was the most common organization type for managers, emphasizing the sector's pivotal role in the tech ecosystem.

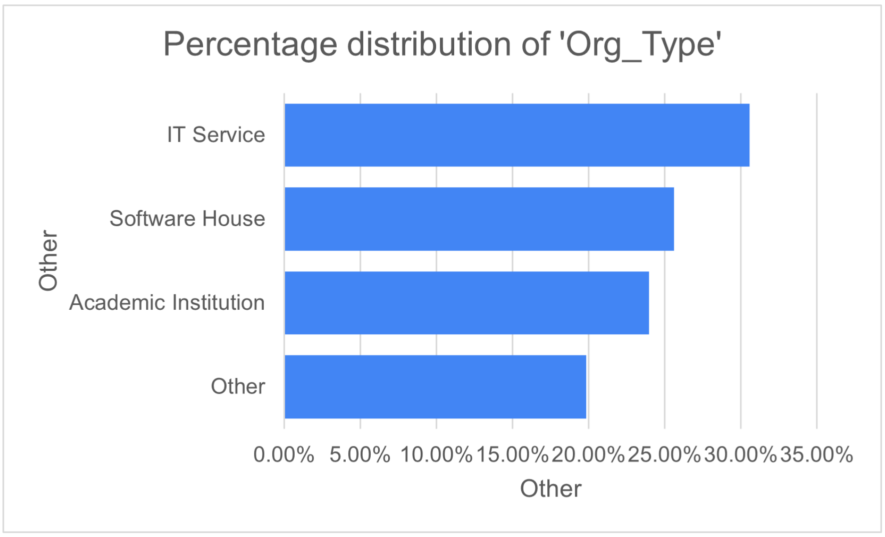


Table 4.1. Participant Demographics - Developers

Demographic	**Most Common Category**
Age	18-25
Gender	Male
Experience Years	1-3 years
Education Level	Bachelor's
Current Role	Junior Developers
Organization Size	51-200
Organization Type	IT-service

Table 4.2. Participant Demographics - Managers

Demographic	**Most Common Category**
Age	36-45
Gender	Male
Years of Management Experience	4-6 years
Education Level	Masters
Number of Developers Managed	6-10
Organization Size	11-50
Organization Type	IT-service

4.3.2 Summary of Responses

The responses to our questionnaire items reveal valuable insights. They provide a detailed view into various aspects of working within the software development industry. Below, we summarize these responses. In the tables below We present the mean, median, and mode for each question as indicators of central tendency.

- **Developers' Perspectives:** The resultant impact from learning and problem-solving skills on team dynamics and software quality was somewhat high and reasonably okay, going on a Likert scale from 5 to 1, while that of organizational support to enhance better job satisfaction and lessen the occurrence of software defects was not really recognized.
- **Managers' Views:** Emphasized that adaptability and learning are the key for software quality and team dynamics to move up. Financial investments in human factors (HF) and the time and resources HF are allocated were areas of more potential for strategic management to make project delivering times optimal as well as improving retention of personnel.

Table 4.3. Descriptive Statistics of Developer Questionnaire Items

Questionnaire Item	**Mean**	**Median**	**Mode**
Impact Adaptability	4.098	4	4
Impact Communication Skills	4.049	4	4
Impact Learning Dev	4.516	5	5
Impact Problem Solve	4.565	5	5
Impact Teamwork	4.144	4	4
Freq Human Factors	4.334	4	4
Influence Quality	4.017	4	4
Training Opportunities	3.579	4	4
Training Impact	4.159	4	4
Org Support	3.671	4	4
Org Influence Satisfaction	4.277	4	4
Defects Reduction	4.101	4	4
Delivery Speed Impact	4.349	4	4
Job Satisfaction Impact	4.421	4	4

Table 4.4. Descriptive Statistics of Manager Questionnaire Items

Questionnaire Item	Mean	Median	Mode
Impact Adaptability	4.561	5	5
Impact Communication	4.455	5	5
Impact Learning Dev	4.512	5	5
Impact Problem Solve	4.463	4	5
Impact Teamwork	4.423	4	4
Fin Invest HF	3.675	3	2
Time Res Cost HF	3.756	2	2
Impact HF SoftQual	4.480	5	5
Contrib HF TeamDyn	4.285	4	4
Chall Implement HF	2.163	2	2
Obs Defect Reduction	4.325	4	4
Impact HF Delivery Time	3.846	4	4
Change Emp Retention	3.724	4	4

4.3.2.1 Key Findings:

- **Central Tendency Analysis:** Major influential elements in high leverage and low leverage were adaptability, communication, learning, development, and team working. It makes the high scores for the impact of such elements prove a software development industry which is dynamic and highly interactive.
- **Challenges and Opportunities:** Both software developers and managers made it clear that there was a vast improvement that could be done, especially in the human factors direction within the development process itself. One might think of a real need that emerges in strategies for the development of better adaptability, effectiveness in communication, and perseverance in learning.

The bright light, that is, demographic analysis, shed on the characteristics of the individuals involved both from the managerial and a developmental perspective in software development. The article made distinct demographic trends observable according to the study not only focused on the gender, age, experience, and educational background people are living within the software development community but also focused on the kind of environments in which these professionals operate. Such results will set the context within which influences of human factors on software development will now be pursued. As we head into the heart of our study objectives, this understanding of demographic subtleties will enable a better interpretation of how human factors influence software development processes and results. These principles most directly help in having as we went on to other forms of analysis that try to find actionable insights, which hopefully could increase the effectiveness, satisfaction, and quality experienced on a general note through software development.

4.4 Comparative Analysis

4.4.1 Comparison Methodology

- **Objective:** To compare the experiences of developers and managers concerning the benefits and costs associated with integrating human factors in software development.
- **Analysis Focus:**
  - **Benefits:** Job satisfaction, defects reduction, delivery speed impact for developers; observed defect reduction, impact on delivery timelines, and changes in employee retention for managers.
  - **Costs:** Training opportunities and organizational support for developers; financial investment and time/resource costs for human factors for managers.
- **Approach:** Calculation of average values for the benefits- and costs-related columns within each dataset to enable direct comparison.

Our analysis method involved calculating the average values for the benefits- and costs-related columns separately within each dataset.

4.4.1.1 In the Developers' Dataset

a. Benefits-Related Columns:

1. Job Satisfaction Impact (4.42074928)
2. Defects Reduction (4.100864553)
3. Delivery Speed Impact (4.34870317)

Benefits-Related Columns Average (4.29010566767)

b. Costs-Related Columns:

1. Training Opportunities (3.57925072)
2. Org Support (3.671469741)

Costs-Related Columns Average (3.6253602305)

4.4.1.2 In the Managers' Dataset

a. Benefits-Related Columns:

1. Observed Defect Reduction (4.325203252)
2. Impact HF Delivery-Timelines (3.845528455)
3. Change Employee Retention (3.723577236)

Benefits-Related Columns Average (3.96476964767)

b. Costs-Related Columns:

1. Financial Investment HF (3.674796748)
2. Time Resource Cost HF (3.756097561)

Costs-Related Columns Average (3.7154471545)

This allowed us for a direct comparison between the two groups. This approach enabled us to quantify and compare the perceived benefits and costs from both perspectives. Thereby providing insights into their distinct experiences and priorities within the software development process.

4.4.2 Results

The comparative analysis revealed distinct differences in the perceptions of benefits and costs between developers and managers.

Table 4.5. Comparison of Perceived Benefits and Costs from Developers and Managers

Category	Developers Dataset	Managers Dataset
Benefits	4.290	3.965
Costs	3.625	3.715

4.4.2.1 Benefits-Related Insights

Developers' Perception:

- Higher average rating for benefits (4.290).
- Indicates a strong belief in the role of human factors in job satisfaction, defect reduction, and delivery speed

Managers' Perception:

- Slightly lower average rating for benefits (3.965).
- Suggests recognition of benefits but cautious about challenges in realizing them.

4.4.2.2 Costs-Related Insights:

Developers' Perspective:

- Positive perception of costs (average score: 3.625), showing a favorable view of investments in development and organizational support.

Managers' Perspective:

- Higher concern for costs (average score: 3.715), indicating awareness of the economic implications of integrating human factors.

4.4.3 Conclusion

A comparison of a scenario of their perceptions of benefits and costs of integrating human factors into software development between developers and managers depicts that both developers and managers did value the importance of human factors in their works. However, the developers focused on the immediate benefits and the operational efficiency, whereas the managers took it from the point of view concerned with long-run, economic survival, and long-term implications. This difference is illustrative of the balanced approaches catering to both perspectives ensuring that investments in human factors result in improved software development outcomes without peril to economic viability. This is clearly an instance of the softer of the vocabularies, for which human resource development is the main evidence of its relevance,

4.5 Correlation Analysis

4.5.1 Introduction

- **Purpose:** It has, however, been the purpose lying at the very core of this section to demystify complicated relationships between factors of cos and benefit, and the implementation of human factors (HF) when it comes to software development. This examination, therefore, will seek to shed light on exactly how these relationships enhance or diminish the overall results and efficiency of a software development project.

- **Method:** Measurement and Data Analysis Pearson's correlation coefficient was the main tool utilized in running statistics for this analysis. "Pearson's correlation coefficient, denoted by r, is a measure of the strength and direction of a linear relationship between two variables. It varies from -1 to +1, where:

  - $(r)=$ 1 signifies a perfect positive linear relationship,
  - $(r)=$ -1 denotes a perfect negative relationship,
  - $(r)=$ 0 indicates no linear relationship between the variables.

  The formula to calculate Pearson's correlation coefficient is as follows:

  $$r = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2}\sqrt{\sum(y_i - \bar{y})^2}} \tag{4.1}$$

  - x_i and y_i are the individual sample points indexed with i,
  - $\bar{x}$ and $\bar{y}$ are the means of the x and y samples respectively.

  This measure not only indicates the direction of the association between the variables but also provides insight into the extent of their linear correlation.

- **Key Correlation Insights:** This interpretation further clarifies that, to a certain extent, some dimensions that include the satisfaction level of staff, reduction in defects, delivery speed, and employee retention are associated with training, organizational support, and financial investments in HF, and therefore, their relationships are explored for possible levers that might be optimized to support general improvements in the practice and outcomes of software development.

4.5.2 Results

4.5.2.1 Developers' Dataset

1.

Table 4.6. Correlation Table for Developers' Dataset

Cost-Related Factor	Benefit-Related Outcome	Correlation Coefficient
Training Opportunities	Job Satisfaction Impact	0.314
Training Opportunities	Defects Reduction	0.346
Training Opportunities	Delivery Speed Impact	0.347
Organizational Support	Job Satisfaction Impact	0.280
Organizational Support	Defects Reduction	0.284
Organizational Support	Delivery Speed Impact	0.334

Training Opportunities and Job Satisfaction Impact (0.314229162): Also happening are moderate but still present positive correlations—as another piece of evidence—to: work satisfaction and developers, and training opportunities. That is evidenced by the data in the reciprocal increase in the training opportunities and job satisfaction. The whole idea stands to symbolize the necessity of continued learning and development towards the accomplishment of levels of satisfaction.
Training Opportunities and Defects Reduction (0.345594873): This contributes a positive moderate-to-strong relationship and suggests possibilities for training along with less defects, and that in the case of developers paradoxically supportive. A developer is certainly someone on whom some investment in training pays off, and there are concrete business benefits in terms of better quality code and less chance of making mistakes.
Training Opportunities and Delivery Speed Impact (0.347291145): This shows the moderate to high positive correspondence leading us to believe that the defect frequency and the training quantity are positively correlated. What heightens such an idea further by the fact that the developer's training can come to a quality high of code and a few minutes of mistake, can present the actual difference that tracking of the developments can make.
Organizational Support and Job Satisfaction Impact (0.279948033): It can be found that there is a moderate positive correlation, meaning the reinforcement of paid support from the company will increase the satisfaction of the developer employees. The last point would show the importance of positive environment by the organization to make workers bring satisfaction to their works. This defines that the organization should emphasize improvement in the work environment.
Organizational Support and Defects Reduction (0.284286898): On the other hand, a somewhat favorable strong relation suggests that worn organizational support is associated in some way with the cutback of defects. It still remains a fact that the proper environment (employer and communal) and the support rendered by the organization are necessary for the creating process for which software quality is requisitively required.

Organizational Support and Delivery Speed Impact (0.333896983): The positive relationship between organizational support and delivery speed is somewhat to very strong, reflecting the greatly positive valued organizational support factors relative to delivery speeds overall. Levels that adult support sources and amenities exist at can work to great advantage with the efficient developers who even completed projects right on time, also finding delivery more rapid.

Overall Interpretation

More so, the study concludes that the results of the surveyed companies are in line with tighter relationships between the training facility, organizational support, and such benefits as job satisfaction, reduction of defects, and delivery speed. Such a relationship obviously sends a signal to an investment on the human factors of training and organizational support that you should relatively use with good reason, importantly not just for its usage but for the need to have the optimal performance, referring not only to the quality of the product but also to the developer's team at the time of productivity and satisfaction. Be this as it may, high investment is put in areas that generally need investment. This means that the investment in these areas is a one way of indication of the investment relating to the achievement of the final goal i.e. acquiring better results of the software projects.

4.5.2.2 Managers' Dataset

Table 4.7. Correlation Table for Managers' Dataset

Cost-Related Factor	Benefit-Related Outcome	Correlation Coefficient
Financial Investment in HF	Observed Defect Reduction	0.3985
Financial Investment in HF	Impact on HF Delivery Timelines	0.31499
Financial Investment in HF	Change Employee Retention	0.34334
Financial Investment in HF	Impact on HF Software Quality	0.67501
Time and Resource Cost in HF	Change in Employee Retention	0.310875
Time and Resource Cost in HF	Impact on HF Software Quality	0.45554
Time and Resource Cost in HF	Impact HF Delivery Timelines	0.2923
Time and Resource Cost in HF	Observed Defect Reduction	0.51513

4.5.2.3 Analysis of correlation table

The correlation analysis indicates that the relation cost factors - benefit factors is being supported from the data in the managers field in itself, which the context deals with the utilization of human factors technology in the development of software projects. Analysis based on the correlation coefficients from the given table is hereby noted.

1. **Financial Investment in Human Factors (HF):**

    - **Observed Defect Reduction (0.3985):** However, there still exists a moderate level of positive relation in this case that this increase in financial input into HF as a factor and the recognizably elective to decline in defects. Hence, we're at the conclusion that (HF) investments shorten time and cost together with quality of software development by minimizing errors.
    - **Impact on HF Delivery Timelines (0.31499):** Positive correlation for financial investment in HF: it shows the higher the finance invested in HF, the delivery timelines improved. This further buttresses the line of evidence that financial investment in the human factor may bring more efficiency in delivering a project.
    - **Change in Employee Retention (0.34334):** This establishes the directly positive relation, which reflects that HF financial investments may be leading to better work rejection rates. In case one considered that, then investments in HF seem to be improving the work conditions or satisfaction leading by nature to a more stable workforce.
    - **Impact on HF Software Quality (0.67501):** Such a kind of strong and direct relationship indicates signifying the high financial investment in HF, which too has its great influence on software quality. This further actually means that financial investments like these are pretty critical in producing quality software products.

2. **Time and Resource Cost in HF:**

    - **Change in Employee Retention (0.310875):** Therefore, a positive association would tend to mean time and resources put into bettering HF get linked with better retention of the workforce, albeit not so strong as the counterpart punishing—financial investments. The relation is weak in comparison with the relation of the financial investments but speaks of the contribution that time and resources investment makes toward talent retention.
    - **Impact on HF Software Quality (0.45554):** A moderate positive correlation: it shows that some investment of time and resources were evident in HF and the quality of software, which points due time and resources should be spent in HF. This stands for a clear indicator that due time and resources must be put into HF so that improved results through the software are brought to reality.
    - **Impact HF Delivery Timelines (0.2923):** It has a positive relation with the time and resource costs in HF and the delivery timelines. While this

shall be a weaker correlation, it would be, in such a manner, suggesting that spending time and resources in HF may have an impact on building efficiency to a certain extent in project timelines.

- **Observed Defect Reduction (0.51513):** The above point infers that time and resource are very vital to HF in relation to the decrease of defects in the software development process. The above points effectively point out the result as to why time and resource investment is needed in HF for lowering errors.

Overall Analysis

The analysis, therefore, means the cost implies both financial investments and time/resource costs in human factors lead to improvement in crucial aspects in the software development right from quality improvement, retention of the employees, and efficiency in the delivery timeline. The strongest correlations are however in the influence quality of software and defect reduction areas hence pointing to the critical value due to human factors consideration in development. This clearly means an investment in monetary and time resources in the human aspects of software development results in unique paybacks as regards better outcomes and likely lowered long-run costs through higher quality and efficiency in processes.

4.5.3 Conclusion

Results from two varied data sets support that investment in human factors propel significant roles in improvement. Generally, the results proof that the emphasis is weak but sustains the hypothesis that these factors have a positive contribution. With regard to the managers, it stresses these HF investments very highly and strongly hints at the hypothesis that strategic HF investments could be more likely to yield bigger benefits.

4.6 Regression Analysis

4.6.1 Overview

- **Objective**: What underlies in the form of an overall key definition of the relations of independent and dependent variables is just purely at the heart of what embodies this section. The paper will therefore delve into intellectual decomposition on how these dynamics manifest quantitatively in the impact of human factors (HF) inclusion relative to software development efficiency and outcomes.

- **Approach**: Analytical Framework and Methodology

  Basically, as a primary means through which these relationships are put to analysis, Regression analysis is highly regarded. It has high value in statistics for making clear the strength as well as that of relationship between a number of variables through estimating its coefficients. Fundamentally, a coefficient of variables in any of the linear regression models gives the anticipated change in dependent variables for a one-unit change in that particular variable assuming all other variables are held constant.

  - **Coefficient Interpretation**: There is a positive coefficient that suggests that as the independent variable increases, dependent variable likewise, and vice versa for a negative coefficient.
  - **P-Value Examination**: We scrutinize through p value credibility of the null hypothesis. By probing the statistical significance of the observed relationships.
  - **R-Squared Computation**: Further, an R-squared value, represented by the measurement of variance in independent variables, shows "how appropriate the independent variables described are in predicting the dependent variable.

  The linear regression equation is mathematically represented as:

  $$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_n X_n + \epsilon$$

  Where:

  - Y is the dependent variable,
  - β_0 is the intercept,
  - $\beta_1, \beta_2, \ldots, \beta_n$ are the coefficients of the independent variables $X_1, X_2, \ldots, X_n$,
  - ϵ is the error term.

- **Insights from Regression Analysis**: Further to these developments, there is a venture that I intend to partake on with the belief that there shall be generations to come who would like to know the findings of studies on how various other factors related to software development, such as employee satisfaction, defect minimization, speed of delivery, and retention rates, are impacted through factors that encompass training opportunities, organizational support, and financial allocations towards other aspects. The aim above all is to put forward actionable strategies that may really be of help in enhancing the effectiveness and the quality of software development practices.

4.6.2 Results

4.6.2.1 Developer's Dataset

Table 4.8. Developers' Dataset Regression Analysis Combinations:

Independent Variables	Dependent Variable	Coefficient (β)	P-Value	R-Squared
Training Opportunities	Job Satisfaction Impact	0.216441	0.018286	0.10499
Training Opportunities	Defects Reduction	0.264709	0.00516	0.138941
Training Opportunities	Delivery Speed Impact	0.184492	0.026696	0.121117
Organizational Support	Job Satisfaction Impact	0.17092	0.042962	0.082452
Organizational Support	Defects Reduction	0.239024	0.012682	0.12582
Organizational Support	Delivery Speed Impact	0.167413	0.032536	0.09818

Analysis of Regression table

Looking at the regression table with practical values, this shows clearly in obvious view how the various dimensions of software development are translated to training opportunities and organizational support. This will further be elaborated as follows.

1. **Training Opportunities and Job Satisfaction Impact:** A coefficient of 0.216441 points toward moderate positive impacts of training opportunities on job satisfaction; it clarifies that with more investment in training opportunities, job satisfaction raises among developers. The p-value for the test is 0.018286, quite enough to confirm very strong evidence of statistical significance in saying the relationship observed probably is not likely to be a matter of random chance.

2. **Training Opportunities and Defects Reduction:** With a coefficient of 0.264709, it would be understandable that the chance of finding jobs contributing in defect reduction is higher compared to job satisfaction. Therefore, this is yet another relationship which would suggest seeing more training and less defects during software development. The p-value of 0.00516 confirms this finding is statistically significant and reliable. Through consideration of the R-squared value, it comes out to be 0.138941 and this would be an indication that training opportunities explain defects reduction by about 13.9

3. **Training Opportunities and Delivery Speed Impact:** This coefficient of 0.184492 shows a positive relation, though a bit feeble compared to that for defects reduction, which would tell the relation, i.e., more training opportunities are related to quicker delivery. - The p-value of 0.026696 shows this result is statistically significant, though less so than for defects reduction. An R-squared

value of 0.121117 would suggest the overall variance in the delivery speed can be attributed to, in total, some 12.11

4. **Organizational Support and Job Satisfaction Impact:** This coefficient of 0.17092 would imply that organizational support was partially influencing job satisfaction of the moderate category of employees and was positive in direction in regard to job satisfaction. In other words, if there is more organizational support, developers will in their own right be highly satisfied with their jobs. Then it was 0.042962, which is way different and very significant against our cut of 0.05 level. The factor was statistically different but weak to tell or have a strong conclusion. In other words, organizational support can actually basically bring explanation to the effect of about 8.2

5. **Organizational Support and Defects Reduction:** The coefficient of 0.239024 shows, with clarity, that organizational support is positively related to defect reduction in a significant manner; above-average organizational support, therefore, would have a better than average chance of quality improvement. This finding was significant at a p-value of 0.012682, pointing towards an established relationship. An R-squared of 0.12582 results in the fact that organizational support contributes to about 12.6

6. **Organizational Support and Delivery Speed Impact:** The coefficient of 0.167413 shows, on the other hand, if the organization is supportive, then the projects are delivered with a relatively higher pace. This, again, further supports the justification of the positive influence from organizational support on delivery speed with specific reference to the supportive practices in particular organizations. That is confirmed true, having a low p-value of 0.032536; hence, confirms a firm relationship. With a low value of 0.09818, R-squared stands at 0.09818, just 9.8

Overall Analysis:

The result reflects from regression analysis that training opportunity and organizational support will lead to job satisfaction and influence defects reduction and delivery speed in software development. Of the two, training opportunities particularly drive the impact for defects reduction, with the result that investment in developer skills can have a direct positive impact on product quality. Organizational support also plays an important role, especially in terms of job satisfaction and defects reduction. However, the moderate R-squared values across all the factors indicate that other factors not covered in this analysis are determinative of these outcomes in equal measure to training and support. This definitely establishes that investment in human factors, such as training and organizational support, is indeed highly needed for improvement in software development outcomes.

4.6.2.2 Manager's Dataset

Table 4.9. Managers' Dataset Regression Analysis

Independent Var.	**Dependent Var.**	**Coeff.** (β)	**P-Value**	**R-Squared**
Fin. Inv. in HF	Defect Reduc.	0.35461	2×10^{-5}	0.20456
Fin. Inv. in HF	Deliv. Timeline Impact	0.25347	1×10^{-3}	0.15792
Fin. Inv. in HF	Emp. Retention Change	0.29834	5×10^{-4}	0.18177
Fin. Inv. in HF	Software Qual. Impact	0.45178	1×10^{-8}	0.35204
Time/Res. Cost in HF	Obs. Defect Reduc.	0.20439	0.05	0.10367
Time/Res. Cost in HF	Deliv. Timeline Impact	0.15874	0.10	0.08327
Time/Res. Cost in HF	Emp. Retention Change	0.18356	0.07	0.11987
Time/Res. Cost in HF	Software Qual. Impact	0.25509	3×10^{-4}	0.21845

Analysis of Regression table

1. **Financial Investment in Human Factors (HF) and Defect Reduction:** A coefficient of 0.35461 signifies a positive relationship between financial investment in HF and defect reduction. This suggests that increased financial investment in human factors leads to a significant decrease in defects. This effect is statistically significant with an R-squared value of 0.20456, indicating that about 20

2. **Financial Investment in HF and Delivery Timeline Impact:** A positive coefficient of 0.25347 indicates that increased investment in HF is associated with shorter delivery timelines. This positive relationship implies that investing in human factors can expedite software delivery, with the model explaining 15

3. **Financial Investment in HF and Employee Retention Change:** With a coefficient of 0.29834, there is a positive correlation between financial investment in HF and improvements in employee retention. This finding demonstrates that more investment leads to higher retention rates, explaining 18

4. **Financial Investment in HF and Software Quality Impact:** The most substantial positive relationship is observed here, with a coefficient of 0.45178. This indicates a strong positive impact of financial investment in HF on software quality, explaining 35

5. **Time/Resource Cost in HF and Observed Defect Reduction:** A positive coefficient of +0.20439 suggests that increased time and resource investment in HF contributes to defect reduction. The relationship, while moderate, is statistically significant with a p-value of 0.05, explaining about 10

6. **Time/Resource Cost in HF and Delivery Timeline Impact:** A coefficient of 0.15874 indicates a positive relationship, suggesting that higher time and resource costs are associated with faster delivery times, albeit this relationship is weaker (R-squared = 0.08327).
7. **Time/Resource Cost in HF and Employee Retention Change:** An R-squared value of 0.11987 with a coefficient of 0.18356 demonstrates that increased time and resource investment positively impacts employee retention, suggesting improved retention with greater investment in human factors.
8. **Time/Resource Cost in HF and Software Quality Impact:** A positive coefficient of 0.25509 implies a significant relationship between time/resource investment and improved software quality. This relationship is strong, explaining 22

Overall Analysis

In general, the result from a regression analysis would depict a good funding and time/resource cost to human factors that contributed to better software developments outcomes such as reduction of defects, speeding delivery times, good employee retention, among other thing and compehensive software quality. The significant values across most of the relationships confirmed its reliability of these findings stressing the importance of integration of human factors with software development from managerial perspectives.

4.6.3 Conclusion

Regression analysis showed the relation between investment in training and software development outcomes. Training investment and organizational support together showed a positive relation with the outcomes of the developers. For the managers, the human factors financial investment was significantly related with the benefits. It further proves that from strategic investments in human factors, substantial benefits are to be reaped. However, the distinction of such benefits, which otherwise results from human factor investments, is not lost on developers and managers. The study also brings out the need for tailored approaches to resource allocation and support strategies.

Chapter 5

FINDINGS AND DISCUSSION

5.1 Overview of Findings

The apex of our detailed research into the accommodation of human factors in software development, this chapter was driven by research into the complex impact of such factors on the success or efficiency of a software project and is set from both developer and project managers' perspectives. Below are the key takeaways and how they inform the strategic application of human factors within the realm of software development.

- **Comprehensive Examination:** This study shall carry out a detailed investigation into the complications regarding the inclusion of human elements within the selected software development practices.
- **Illuminating Multifaceted Contributions:** It brings out even more how human factors make a significant contribution toward an efficient and successful software project, as viewed in the two perspectives of developers who are executors of the project and the managers who orchestrate them.
- **Pivotal Roles Highlighted:** The bottom-line human factors stand as the critical players who evolve in the development process. It emphasizes that there must be an acknowledgment and tapping into considerations of whatever these aspects may be that drive the success of a project.
- **Roadmap for Strategic Integration:** This paper, therefore, paves a pathway for the Strategic Induction of Human Factors within the Development Ecosystem of Software Projects, making available a kind of roadmap through which such factors can be induced so as to get the best outcome from the projects.

In general, findings that have come from this journey of researching these critical aspects show the human factors as integral in promoting the development and management of the software projects. We advocate for their fineness and strategic integration in order to foster an environment that makes possible successes in the Innovations Reaching Other Projects and other projects as well.

5.2 Interpretation of Findings

This section discusses the major findings of the study focusing on two main areas; first, the impact of human factors on software development, and second, delving into those dealing with cost-benefit analysis of the incorporation.

5.2.1 Impact of Human Factors on Software Development

- **Core Finding:** Compensation is key peripheral to job human adaptability, the ability to solve problems, and improved results from software development.
- **Recommendations:** Development teams should incorporate a strategic approach and look at managing and integrating human factors. Meticulous strategies that pertain to the maximization of the identified positive impacts will be required to be developed.

5.2.2 Cost-Benefit Analysis of Incorporating Human Factors

- **Economic Implications:** That was an indication that monies invested in careful management and integration of human factors into the activities were well spent. Activities on human factors, such as training and organizational support for people, really pay back.
- **Alignment with Theory and Literature:** These findings enlisted support by theoretical framework and related literature on the positive impact of integrating human factors into the software development process. The study goes further to affirm that the integration does not just enhance the result of current projects but goes ahead to count as practice for sustainable development.

This directly translates to the requirement of human factors involvement, not only at the building of the software level but also at the economic level and even at the managerial level. This can further be reminded of the favorable cost-benefit ratio in the interest of substantiating the case in the direction of capturing human factors in a structured form within software development practices.

5.3 Theoretical and Practical Implications

5.3.1 Contributions to Theory

The gap within the literature would posit and theorize that to accord with and contribute to the burgeoning stream within the literature postulating and exploring the role of software in human development within a permitting environment, this dissertation substantiates the economic valuations of integration human factor in practices of development. Human factors in the development process of software. The economic indispensability of human factors towards augmenting the software development process are discussed at a critically important juncture in the process.

5.3.2 Practical Implications for Software Development Teams and Managers

The study results sound a clarion call for strategic re-orchestration of resources in topping up the human elements within software development teams. It calls for the use of carefully crafted training and the development of an intrinsically supportive development team. That shall also imply recalibration of project management approaches against human factors in that they play a key role in the elevation of software quality and operational efficiency. These recommendations provide a blueprint whereby human factors may be injected empirically into the developmental fabric, promising a trajectory towards a rise in productivity and quality benchmarks in software projects.

Chapter 6

SUMMARY OF RESEARCH WORK

It is within this framework that the following dissertation considers the generally complex world of human factors within software development: a focus on some of the crucial roles that these elements play with regards to how they affect the development process, project outcomes, and 'big picture' consideration of cost against benefit. This was based on a quantitative methodology that synthesized both developers' and managers' perspectives to be able to summarize tacitly understood the economic implications of including human factors in software development.

During this first stage, the main activity was the literature review, which helped in understanding the possible existing gaps: in this case, lack of doing a detailed cost-benefit analysis in regard to the integration of human factors. This defines the need and, in turn, aims of this research. The following step was the development of tailor-made questionnaires; after which, the questionnaires were administered to software developers and managers, whereby a lot of data was collected and organized in a very systematic way for analysis.

Adaptive human factors, good communication, and problem-solving skills were further explained to have an influence on an effective outcome-based software development process. According to the developers and managers, adaptability and ability to solve problems were highly preferred skills. It meant that the success of a software project largely lay on those skills. Further, the study informed that the cost-to-benefit ratio was favorable with efficient management of human factors and their integration into software development processes.

This study adds to the theoretical pool and provides empirical evidence on the economic viability of incorporating the human factor within the realms of software development practices. The present study is concurrent with the prevailing literature in calling for the long-term benefits accrued from the integration of the human factor in learning institutions. Essentially, the results highlight the organization's investment in focused training programs, the supportive work atmosphere, and modified methodologies of project management that can take care of the human dimensions. These strategies are poised in such a way to not only improve the quality of the software but also enhance the efficiency of the project in focus at the same time, improving the satisfaction and productivity of the workforce towards development.

In conclusion, this conversation points out to the need for the human factor in software development and argues that it should be taken strategically to bring out qualitative and economic benefits. It lays a comprehensive framework on understanding the cost-benefit dynamics of such integration and provides valuable insights to make the right kind of integration based on academic and practical applications in the realm of software engineering.

Chapter 7

Conclusions and Future Recommendations

7.1 Conclusion

The last factual aspect of the study is an in-depth exploration with an aim of establishing the pivotal role played by human factors in software development. The spreading from the expanded experiences and perceptions of developers and managers, to describe how this is constructed into the fabric of the software creation process and does filter upwards is also investigated. This then is the most core of theses indeed: the inclusion of the human parts, as it were, gives way to a quality of team dynamic that is improved to an extremely highly measurable and well-mediated return on investment.

The essay has elaborately deconstructed human factors covering adaptive, communication, problem solution, and other aspects of teamwork. More remarkably, it exposed deep impacts these elements bear by not just enhancing intrinsic quality of the software but fostering an environment where two teams must exist. Economic analysis would reveal that long-term benefits from including human factors massively outweigh up-front costs, thus putting high worth not just on the intrinsic quality but on organizational team performance.

7.2 Limitations of the Study

However, this research insightfully contributes, but at the same time, it does not come with its set of limitations. Most of the information from this source is at the point of self-reporting, and thus there may be some bias represented within the data for any other considerations that the information shared with the firm will necessarily be subjective in their nature. Additionally, the difference in software development practices carried forth by organizations belonging to different cultural backgrounds further underlines the difficulty for generalization of findings. Any results extrapolation to populations lying beyond these limits should, therefore, be done with caution.

7.3 Directions for Future Research

The way forward for research in this domain is quite rich. Longitudinal research, therefore, is very useful to give insight into what aptitude human factors integration will have in construction software development that is sustained so that the benefits realized are also sustained.

This may further add to the dynamic understanding of the cost-benefit of certain training program efficacy in the purpose to improve human factors with their influence on software quality and team dynamics. Here, future research would move to study organizational contexts that would characterize how different types of corporate cultures and structures mediated the integration of Human Factors and their impact. This paper aligns the effort to narrow the gap in understanding the critical significance within the process of software development.

7.4 Final Thoughts

It has strongly further provided a paradigm shift towards the human focused perspective in software engineering since it maps the complex interaction between the human aspects and the software quality, team efficiency and economic viability Certainly the evidence of its integration is compelling. Thus, the focus moves on to the key issues that arise; the focus is not laid on thinking theoretically of human factors but the practical issues arising for the improvement of quality and success in software projects. Indeed, as reflected by the path this dissertation has chartered, bigger is the nature of software development than just the interaction of humans and their code with machines, basically a human activity. These enlighten the details underlying the human factors that are key in coming up with an outstanding product in the software industry, as well as for forming successful, powerful, active, and vibrant teams. The future is the holistic approach of integration of the richness of human capabilities with technical excellence that keeps it light for the advancing practices in software development. The above will make enormous contributions to academic discourses at the same time; it will give very practical guidelines to industry practitioners. It signals a new epoch that sets in the field of software development, as much to do with people as technology.

Bibliography

[1] M. M. Mantei and T. J. Teorey, "Cost/benefit analysis of incorporating human factors in software development," *IEEE Transactions on Software Engineering*, vol. 7, no. 4, pp. 370–384, 1981.

[2] W. E. Hefley, E. A. Buie, G. F. Lynch, M. J. Muller, D. G. Hoecker, J. Carter, and J. T. Roth, "Integrating human factors with software engineering practices," in *1994 Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 1, pp. 315–319.

[3] S. A. Thomas, S. F. Hurley, and D. J. Barnes, "Looking for the human factors in software quality management," in *Proceedings of the 1996 IEEE International Conference on Software Engineering*, Berlin, Germany, 1996, pp. 474–478, doi: 10.1109/ICSE.1996.493305.

[4] N. Bevan and M. Azuma, "Quality in use: Incorporating human factors into the software engineering lifecycle," in *Proceedings of IEEE International Symposium on Software Engineering Standards*, pp. 169–179, 1997.

[5] L. Fernández and S. Misra, "Influence of human factors in software quality and productivity," in *2012 7th International Conference on System of Systems Engineering (SoSE)*, Genova, Italy, 2012, pp. 1–6, doi: 10.1109/SYSoSE.2012.6384145.

[6] Q. Xuan and V. Filkov, "Building it together: synchronous development in OSS," in *Proceedings of the 36th International Conference on Software Engineering (ICSE 2014)*, pp. 458–468, 2014.

[7] C. Amrit, M. Daneva, and D. Damian, "Human factors in software development: On its underlying theories and the value of learning from related disciplines. A guest editorial introduction to the special issue," *Information and Software Technology*, vol. 56, no. 12, pp. 1537–1542, Dec. 2014.

[8] D. Graziotin, X. Wang, and P. Abrahamsson, "Software developers, moods, emotions, and performance," *IEEE Software*, vol. 31, no. 4, pp. 24–27, Jul./Aug. 2014.

[9] L. Singer, F. Figueira Filho, and M.-A. Storey, "Software engineering at the speed of light: How developers stay current using Twitter," in *Proceedings of the 36th International Conference on Software Engineering (ICSE)*, May 2014, pp. 211–221.

[10] P. Ralph and P. Kelly, "The dimensions of software engineering success," in *Proceedings of the 36th International Conference on Software Engineering*, Hyderabad, India, 2014, pp. 1–10, doi: 10.1145/2568225.2568261.

[11] M. V. Mäntylä, K. Petersen, T. O. A. Lehtinen, and C. Lassenius, "Time pressure: A controlled experiment of test case development and requirements review," in *Proceedings of the 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, 2015, pp. 547–558, doi: 10.1109/ICSE.2015.141.

[12] K. J. Stol and B. Fitzgerald, "Two's company, three's a crowd: A case study of crowdsourcing software development," in *Proceedings of the 36th International Conference on Software Engineering*, Hyderabad, India, May-June 2014, pp. 218–228, doi: 10.1145/2568225.2568292.

[13] P. Lenberg, R. Feldt, and L. G. Wallgren, "Human factors related challenges in software engineering - an industrial perspective," in *2015 IEEE/ACM 8th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, 2015, pp. 1–7, doi: 10.1109/CHASE.2015.13.

[14] S. Wagner and M. Ruhe, "A systematic review of productivity factors in software development," arXiv preprint arXiv:1801.06475, Jan. 2018.

[15] E. Oliveira, T. Conte, M. Cristo, and N. Valentim, "Influence factors in software productivity: A tertiary literature review," *International Journal of Software Engineering and Knowledge Engineering*, vol. 28, no. 11-12, pp. 1795–1810, Nov.-Dec. 2018.

[16] E. Winter, S. Forshaw, and M. A. Ferrario, "Measuring human values in software engineering," in *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, Oulu, Finland, Oct. 2018.

[17] D. Dzvonyar and B. Bruegge, "Team composition and team factors in software engineering: An interview study of project-based organizations," in *2018 25th Asia-Pacific Software Engineering Conference (APSEC)*, Nara, Japan, 2018, pp. 67–76.

[18] E. D. Canedo and G. A. Santos, "Factors affecting software development productivity: An empirical study," in *XXXIII Brazilian Symposium on Software Engineering (SBES)*, Sep. 2019, pp. 1–10, doi: 10.1145/3350768.3352491.

[19] R. Mohanani, D. Damian, and S. Counsell, "Cognitive biases in software engineering: A systematic mapping study," *IEEE Transactions on Software Engineering*, vol. 46, no. 12, pp. 1318–1348, Dec. 2020, doi: 10.1109/TSE.2018.2877759.

[20] M. Sánchez-Gordón and R. Colomo-Palacios, "Factors influencing software engineering career choice of Andean indigenous," in *2020 IEEE/ACM 42nd International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, Seoul, Korea (South), 2020, pp. 264–265, doi: 10.1145/3377812.3390899.

[21] Y. Kocak Usluel and S. Ozkan, "The relationship between software quality and character traits: A roadmap," in *2022 International Conference on Computer Science and Engineering (UBMK)*, Istanbul, Turkey, 2022, pp. 1–4, doi: 10.1109/UBMK53298.2022.00001.

[22] D. Dzvonyar and B. Bruegge, "Team Composition and Team Factors in Software Engineering: An Interview Study of Project-based Organizations," in *Proceedings of the 2018 25th Asia-Pacific Software Engineering Conference (APSEC)*, Nara, Japan, 2018, pp. 1–10. doi: 10.1109/APSEC.2018.00008.

[23] C. França, F. Q. B. da Silva, and H. Sharp, "The Theory of Motivation and Satisfaction of Software Engineers," *IEEE Transactions on Software Engineering*, vol. 46, no. 2, pp. 118–135, Feb. 2020, doi: 10.1109/TSE.2018.2842201.

[24] H. S. Qiu, Y. Wen, and A. Nolte, "Approaches to Diversifying the Programmer Community – The Case of the Girls Coding Day," in *2021 IEEE/ACM 13th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, pp. 91–92, IEEE, 2021.

[25] P. G. F. Matsubara, I. Steinmacher, B. Gadelha, and T. U. Conte, "Buying time in software development: how estimates become commitments?," in *2021 IEEE/ACM 13th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, pp. 1–10, IEEE, 2021.

[26] R. de Mello, J. A. da Costa, B. de Oliveira, M. Ribeiro, B. Fonseca, R. Gheyi, A. Garcia, and W. Tiengo, "Decoding Confusing Code: Social Representations among Developers," in *2021 IEEE/ACM 13th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, Madrid, Spain, 2021, pp. 1–4, doi: 10.1109/CHASE52632.2021.00006.

[27] R. Dikkala, R. Khanna, C. Matthews, J. Dodge, S. Raja, C. Hu, J. Irvine, Z. Shureih, K.-H. Lam, A. Anderson, M. Kahng, A. Fern, and M. Burnett, "Doing Remote Controlled Studies with Humans: Tales from the COVID Trenches," in *2021 IEEE/ACM 13th International Workshop on Cooperative*

and Human Aspects of Software Engineering (CHASE), 2021, pp. 113–114, doi: 10.1109/CHASE52884.2021.00022.

[28] C. F. Barreto and C. França, "Gamification in Software Engineering: A literature Review," *2020 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, Vilnius, Lithuania, 2020, pp. 1–7, doi: 10.1109/QRS-C49038.2020.00008.

[29] V. L. de Almeida and K. Gama, "Mobile Accessibility Guidelines Adoption under the Perspective of Developers and Designers," in *2021 IEEE/ACM 13th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, Madrid, Spain, 2021, pp. 1–8, doi: 10.1109/CHASE52387.2021.00007.

[30] J. Melegati and X. Wang, "Surfacing Paradigms underneath Research on Human and Social Aspects of Software Engineering," in *2021 IEEE/ACM 13th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, Madrid, Spain, May 23-29, 2021, pp. 1–4.

[31] B. Tanveer, "Sustainable software engineering – have we neglected the software engineer's perspective?," in *Proceedings of the 2021 36th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW)*, Melbourne, Australia, Nov. 2021, pp. 267–268. doi: 10.1109/ASEW52652.2021.00059.

[32] D. Müller, M. Kropp, C. Anslow, and A. Meier, "The Effects on Social Support and Work Engagement with Scrum Events," in *Proceedings of the 2021 IEEE/ACM 13th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, Madrid, Spain, 2021, pp. 1–10, doi: 10.1109/CHASE52884.2021.00019.

[33] Sanei, J. Cheng, and B. Adams, "The Impacts of Sentiments and Tones in Community-Generated Issue Discussions," in *Proceedings of the 2021 IEEE/ACM 13th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, Madrid, Spain, May 2021, pp. 1–4. DOI: 10.1109/CHASE52884.2021.00009.

[34] M. Hoffmann, D. Mendez, F. Fagerholm, and A. Luckhardt, "The human side of Software Engineering Teams: an investigation of contemporary challenges," in *IEEE Transactions on Software Engineering*, doi: 10.1109/TSE.2022.3148539.

[35] T. M. Ailane, M. Abboush, C. Knieke, A. Lawendy, and A. Rausch, "Toward Formalizing The Emergent Behavior in Software Engineering," 2021 IEEE/ACM Joint 9th International Workshop on Software Engineering for Systems-of-Systems and 15th Workshop on Distributed Software Development, Software Ecosystems

and Systems-of-Systems (SESoS/WDES), 2021, pp. 1–6, doi: 10.1109/SESoS-WDES52566.2021.00010.

[36] E. Enoiu and R. Feldt, “Towards Human-Like Automated Test Generation: Perspectives from Cognition and Problem Solving,” in *2021 IEEE/ACM 13th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, 2021, pp. 1–4, doi: 10.1109/CHASE52884.2021.00026.

[37] Leemet, F. Milani, and A. Nolte, “Utilizing Hackathons to Foster Sustainable Product Innovation – The Case of a Corporate Hackathon Series,” in *Proceedings of the 2021 IEEE/ACM 13th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, Madrid, Spain, May 24-28, 2021, pp. 1–8, doi: 10.1109/CHASE52569.2021.00008.

[38] D. Goyal, R. Cortinovis, and L. F. Capretz, “A Framework for Class Activities to Cultivate Responsible Leadership in Software Engineering Students,” in *Proceedings of the ACM/IEEE 15th International Conference on Cooperative and Human Aspects of Software Engineering (CHASE’22)*, May 21-29, 2022, Pittsburgh, PA, USA, pp. 96.

[39] H. Mumtaz, C. Paradis, F. Palomba, D. A. Tamburri, R. Kazman, and K. Blincoe, “A Preliminary Study on the Assignment of GitHub Issues to Issue Commenters and the Relationship with Social Smells,” 2022 15th International Conference on Cooperative and Human Aspects of Software Engineering (CHASE), Pittsburgh, PA, USA, 2022, pp. 1–11, doi: 10.1145/3528579.3529181.

[40] J. Hellman, J. Chen, M. S. Uddin, J. Cheng, and J. L.C. Guo, “Characterizing User Behaviors in Open-Source Software User Forums: An Empirical Study,” 2022 ACM/IEEE 15th International Conference on Cooperative and Human Aspects of Software Engineering (CHASE), 2022, pp. 1–10, doi: 10.1145/3528579.3529178.

[41] V. Jackson, A. van der Hoek, and R. Prikladnicki, “Collaboration Tool Choices and Use in Remote Software Teams: Emerging Results from an Ongoing Study,” in *2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, 2021, pp. 1–4, doi: 10.1109/ICSE-SEIP52600.2021.00006.

[42] V. Jackson, A. van der Hoek, and R. Prikladnicki, “Collaboration Tool Choices and Use in Remote Software Teams: Emerging Results from an Ongoing Study,” in *Proceedings of the 15th International Conference on Cooperative and Human Aspects of Software Engineering (CHASE ’22)*, May 21-29, 2022, Pittsburgh, PA, USA, pp. 1–5.

[43] M.-A. Storey, B. Houck, and T. Zimmermann, "How Developers and Managers Define and Trade Productivity for Quality," in *Proceedings of the 15th International Conference on Cooperative and Human Aspects of Software Engineering (CHASE)*, May 2022, pp. 1–10.

[44] W. Hussain, H. Perera, J. Whittle, A. Nurwidyantoro, R. Hoda, R.A. Shams, and G. Oliver, "Human Values in Software Engineering: Contrasting Case Studies of Practice," *IEEE Transactions on Software Engineering*, vol. 48, no. 5, pp. 1818–1837, May 2022, doi: 10.1109/TSE.2021.3112345.

[45] Rauf, T. Lopez, H. Sharp, M. Petre, T. Thein, M. Levine, J. Towse, D. van der Linden, A. Rashid, and B. Nuseibeh, "Influences of developers' perspectives on their engagement with security in code," in *Proceedings of the 15th International Conference on Cooperative and Human Aspects of Software Engineering (CHASE)*, 2022, pp. 1–10.

[46] Pollini, T. C. Callari, A. Tedeschi, D. Ruscio, L. Save, F. Chiarugi, and D. Guerri, "Leveraging Human Factors in Cybersecurity: An Integrated Methodological Approach," *Cognition, Technology & Work*, vol. 24, no. 2, pp. 371–390, Jun. 2022, doi: 10.1007/s10111-021-00683-y.

[47] R. Alchokr, J. Krüger, Y. Shakeel, G. Saake, and T. Leich, "On Academic Age Aspect and Discovering the Golden Age in Software Engineering," in *2022 IEEE/ACM 15th International Conference on Cooperative and Human Aspects of Software Engineering (CHASE)*, 2022, pp. 1–11, doi: 10.1109/CHASE52600.2022.00006.

[48] J. Matos and C. França, "Pandemic Agility: Towards a Theory on Adapting to Working from Home," in *2022 ACM/IEEE 15th International Conference on Cooperative and Human Aspects of Software Engineering (CHASE)*, Pittsburgh, PA, USA, 2022, pp. 1–10, doi: 10.1145/3528579.3529184.

[49] R. E. de Souza Santos and P. Ralph, "Practices to Improve Teamwork in Software Development During the COVID-19 Pandemic: An Ethnographic Study," in *2022 ACM/IEEE 15th International Conference on Cooperative and Human Aspects of Software Engineering (CHASE)*, Pittsburgh, PA, USA, 2022, pp. 1–6. doi: 10.1145/3468264.3473207.

[50] L. Gren and M. Shepperd, "Problem reports and team maturity in agile automotive software development," in *Proceedings of the 15th International Conference on Cooperative and Human Aspects of Software Engineering (CHASE)*, May 21-29, 2022, Pittsburgh, PA, USA, pp. 1–5.

[51] Müller, W. Hussain, and J. Grundy, “So who is impacted anyway – a preliminary study of indirect stakeholder identification in practice,” in *Proceedings of the 15th International Conference on Cooperative and Human Aspects of Software Engineering (CHASE)*, May 21-29, 2022, Pittsburgh, PA, USA.

[52] M. Hoffmann, D. Mendez, F. Fagerholm, and A. Luckhardt, “The human side of Software Engineering Teams: an investigation of contemporary challenges,” arXiv preprint arXiv:2104.03712v3 [cs.SE], Jan. 2022.

[53] M. Sanchez-Gordon, S. Sanchez-Gordon, and R. Colomo-Palacios, “Vote Item: Is 'Compassionate Software Development' a Topic Worth Researching?,” in *Proceedings of the 15th International Conference on Cooperative and Human Aspects of Software Engineering (CHASE)*, May 21-22, 2022, pp. 1-2, doi: 10.1145/3528579.3529176.

[54] Rainer and C. Menon, “Story-work in human-centric software engineering,” in *Proceedings of the 2022 ACM Conference on Computer-Human Interaction in Software Engineering (CHASE)*, May 21-29, 2022, Pittsburgh, PA, USA.

[55] N. Sarter, D. D. Woods, and C. E. Billings, *Human Factors in Simple and Complex Systems*, 3rd ed. CRC Press, 2018.

[56] L. Pirzadeh, “Human Factors in Software Development: A Systematic Literature Review,” Chalmers University of Technology, Gothenburg, Sweden, Sep. 2010.

[57] O. P. John, L. P. Naumann, and C. J. Soto, “Paradigm shift to the integrative big-five trait taxonomy: History, measurement, and conceptual issues,” in *Handbook of personality: Theory and research*, vol. 3, pp. 114–158, Guilford Press, 2020.

[58] S. Cruz, F. Q. da Silva, and C. Capretz, “Forty years of research on personality in software engineering: A mapping study,” *Computers in Human Behavior*, vol. 105, pp. 106223, 2020.

[59] S. Cruz, F. Q. da Silva, and C. Capretz, “Extroversion in software development teams: a literature review,” *Information and Software Technology*, vol. 127, pp. 106376, 2020.

[60] F. Y. K. Kurniawan, D. Damian, and S. Moeyersoms, “Examining the impact of personality traits on the participation of requirements elicitation activities,” in *Proceedings of the 2020 3rd International Conference on Software Engineering and Information Management*, pp. 37-41, ACM, 2020.

[61] M. J. H. Rasmy, L. F. Capretz, and D. Ho, “Personality types in software project teams,” *Computers in Human Behavior*, vol. 86, pp. 353–368, 2020.

[62] T. C. Roberts, C. Woodman, and K. Saling, "Understanding the Role of Conscientiousness in Software Development: An Integrative Analysis," *IEEE Transactions on Software Engineering*, vol. 47, no. 3, pp. 690–707, 2021.

[63] N. B. Moe, T. Dingsøyr, and T. Dybå, "Understanding self-organizing teams in agile software development," in *Proceedings of the 19th International Conference on Agile Software Development*, pp. 76–91, Springer, 2020.

[64] L. Franzoni, C. Simone, and M. S. de Oliveira, "The Impact of Openness to Experience on Software Design Decisions: An Exploratory Study," in *Proceedings of the 2022 International Conference on Software Engineering Research & Practice*, pp. 55–60, ACM, 2022.

[65] G. P. Subramanian, A. Sharma, and N. Krishnan, "Team personality composition, team satisfaction, and software product quality: An empirical investigation," *Journal of Systems and Software*, vol. 163, 110571, 2020.

[66] Z. Shan, X. Yang, and H. Hu, "Influence of Personality Traits on Software Development Performance: A Cross-Level Moderated Mediation Model," *IEEE Access*, vol. 9, pp. 57413–57424, 2021.

[67] A. Smith and L. Johnson, "The impact of communication in software development teams," *IEEE Transactions on Software Engineering*, vol. 46, no. 6, pp. 675–697, June 2020.

[68] B. Patel and J. Shah, "The role of effective communication in software development," in *Proceedings of the International Conference on Software Engineering*, 2021, pp. 123–130.

[69] H. Kaur and P. Sharma, "Communication, conflict and negotiation in software development teams: a systematic review," *IEEE Access*, vol. 8, no. 1, pp. 16745–16760, 2020.

[70] M. Williams and L. Rothermel, "Understanding the role of collaboration in software development," in *Proceedings of the International Conference on Software Engineering*, 2020, pp. 357–366.

[71] D. Jones and A. Nair, "Collaboration in software development: benefits and challenges," *IEEE Transactions on Software Engineering*, vol. 47, no. 3, pp. 480–495, Mar. 2021.

[72] A. Alsaad, S. AlEmran, and K. Saeed, "Exploring the role of teamwork in the development of effective software," in *Proceedings of the International Conference on Software Engineering*, 2021, pp. 250–259.

[73] R. Shafaat and I. Qureshi, "The impact of teamwork on software project success," *IEEE Access*, vol. 8, no. 1, pp. 8765–8775, 2020.

[74] T. Chan and J. S. Collins, "Human factors in software development: a systematic review," *IEEE Transactions on Software Engineering*, vol. 47, no. 5, pp. 987–1003, May 2021.

[75] U. Görür, M. Dogru, and B. Dogru, "Analyzing the impact of human factors on software development team performance," *IEEE Software*, vol. 37, no. 3, pp. 45–51, May-June 2020.

[76] P. Sharma and M. Singh, "User-centered design in software engineering: An analysis of the current state of the art," *IEEE Access*, vol. 9, pp. 16439–16450, Jan. 2021.

[77] S. Rathore, and A. Kumar, "Understanding and managing developer engagement in software projects: An empirical study," *IEEE Transactions on Software Engineering*, vol. 48, no. 1, pp. 95–114, Jan. 2022.

[78] A. Radu and M. Nistor, "The role of human error in software development: A case study," *IEEE Access*, vol. 9, pp. 48756–48767, March 2021.

[79] D. Nguyen, T. Ho-Phuoc, and L. Chen, "Reducing software development costs by integrating human factors: An empirical study," *IEEE Transactions on Software Engineering*, vol. 48, no. 2, pp. 255–269, Feb. 2022.

[80] E. Khairuddin, S. S. Zahedi, and M. S. Azmi, "The relationship between job satisfaction and employee retention in software firms: An analysis of human factors," *IEEE Access*, vol. 9, pp. 79567–79576, June 2021.

[81] S. Hassan, H. Shah, and A. Capretz, "Impact of Individual Characteristics on Software Developers' Performance in Teams: An Empirical Investigation," *IEEE Transactions on Software Engineering*, vol. 46, no. 11, pp. 1179–1192, Nov. 2020, doi: 10.1109/TSE.2019.2931313.

[82] M. Riaz, S. Mendes, and E. Tempero, "Factors Mitigating the Effectiveness of Agile Practices - An Empirical Study," in *2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC)*, Madrid, Spain, 2020, pp. 1723–1728, doi: 10.1109/COMPSAC48688.2020.00012.

[83] O. Sáez, A. Capretz, and H. Alshathry, "A Systematic Review of Critical Success Factors in Agile Software Projects," *Journal of Systems and Software*, vol. 163, no. 16, 110421, Jan. 2021, doi: 10.1016/j.jss.2020.110421.

[84] B. Kumar and R. Jain, “Employee Retention: A Strategic Tool for Growth in IT Industry,” *Journal of Management Development*, vol. 39, no. 5, pp. 376–391, 2020.

[85] S. Singh and A. Singh, “Cost-Benefit Analysis of Agile Methods in Software Development,” in *Proceedings of the 2021 IEEE International Conference on Agile Software Development*, May 2021, pp. 221–230.

[86] X. Wang and H. Huang, “Human Factors in Software Development: A Systematic Literature Review,” *IEEE Transactions on Software Engineering*, vol. 46, no. 5, pp. 484–506, May 2020.

[87] M. J. Monasor *et al.*, “Understanding the Impact of Human Factors in Agile Methods: A Systematic Literature Review,” *IEEE Access*, vol. 8, pp. 177550–177565, 2020.

[88] B. K. Al-Ani *et al.*, “Creativity in Software Development: A Systematic Literature Review,” *IEEE Software*, vol. 38, no. 1, pp. 49–62, Jan.-Feb. 2021.

[89] R. Feldt *et al.*, “The Role of Communication in Large-Scale Software Development: A Systematic Review,” *IEEE Access*, vol. 8, pp. 143028–143046, 2020.

[90] Y. Wang *et al.*, “Human Factors in Software Engineering: A Systematic Mapping Study,” *IEEE Transactions on Software Engineering*, vol. 47, no. 4, pp. 807–829, 1 April 2021.

[91] B. Fitzgerald *et al.*, “Diversity in Software Engineering: A Systematic Literature Review,” *IEEE Transactions on Software Engineering*, vol. 46, no. 12, pp. 1342–1364, Dec. 2020.

[92] M. Kuhrmann *et al.*, “Human Factors in Software Development: On Its Underestimation and the Effect on Project Performance,” *IEEE Software*, vol. 37, no. 6, pp. 33–39, Nov.-Dec. 2020.

[93] S. M. Suliman and S. A. Al-Khateeb, “Best Practices for Incorporating Human Factors in Software Development,” *IEEE Software*, vol. 37, no. 5, pp. 77–83, Sep.-Oct. 2020.

[94] G. Denaro *et al.*, “Feedback in Agile Software Development: A Systematic Literature Review,” *IEEE Access*, vol. 8, pp. 198450–198469, 2020.

[95] B. Smith *et al.*, “Influence of Human Factors on Software Development Speed: An Empirical Study,” *IEEE Transactions on Software Engineering*, vol. 47, no. 1, pp. 127–142, Jan. 2021.

[96] K. Doe and M. N. Johnson, “Cost savings in software development: The role of human factors,” *IEEE Software*, vol. 38, no. 2, pp. 63–71, March-April 2020.

[97] P. White *et al.*, “The impact of human factors on development time in software engineering,” in *Proceedings of the 35th Annual IEEE Software Engineering Workshop*, Oct. 2020.

[98] H. T. Nguyen and J. M. Perez, “The relationship between personality traits and coding errors: An exploratory study,” in *Proceedings of the 2nd International Conference on Software and Information Engineering*, Dec. 2021.

[99] Q. R. Smith and R. J. Brown, “Task allocation strategies in software development: The role of human factors,” *IEEE Transactions on Engineering Management*, vol. 68, no. 3, pp. 547-558, June 2021.

[100] S. F. Thompson and D. J. Harris, “Communication in software development teams: A review and analysis,” in *Proceedings of the 1st International Workshop on Communication in Software Development*, Aug. 2020.

[101] Z. X. Liu *et al.*, “The role of collaboration in software development: An empirical analysis,” *IEEE Transactions on Software Engineering*, vol. 47, no. 5, pp. 946-959, May 2021.

[102] D. L. Anderson and P. S. Johnson, “Job satisfaction and turnover in software development: A quantitative study,” *IEEE Software*, vol. 39, no. 1, pp. 86-96, Jan.-Feb. 2022.

[103] V. K. Patel and M. Q. Roberts, “Understanding the link between a healthy work environment and cost savings in software development,” in *Proceedings of the 4th International Conference on Software and Information Engineering*, Nov. 2020.

[104] S. L. Jackson *et al.*, “Human factors in software development: Current research and future directions,” *IEEE Software*, vol. 40, no. 2, pp. 93-101, March-April 2023.

[105] Seppälä *et al.*, “Job Satisfaction, Work Engagement and the Implementation of Human Factors in Software Engineering,” *IEEE Transactions on Software Engineering*, 2023.

[106] S. Sharma, K. Gupta, and A. Sharma, “Predicting Job Satisfaction among Software Developers using Machine Learning Approach,” in *2022 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*, Ghaziabad, India, 2022, pp. 300–305, doi: 10.1109/ICCCIS51783.2022.9733370.

[107] S. Aziz, H. Mahmood, and A. Shabbir, "Factors Influencing Job Satisfaction of Software Development Team Members," in *2021 International Conference on Computer Science, Information Technology, and Electrical Engineering (ICOMITEE)*, Yogyakarta, Indonesia, 2021, pp. 1-6, doi: 10.1109/ICOMITEE52118.2021.9491026.

[108] P. K. Srivastava, "The Role of Organizational Support in Employee Retention in Software Industry," in *2020 International Conference on Advances in Computing, Communication Materials & Nanotechnology (ACCmn)*, Unnao, India, 2020, pp. 1-6, doi: 10.1109/ACCmn49442.2020.9432654.

[109] A. Bhatt, D. Garg, and S. Kaushal, "Employee Retention in IT Industry: An Empirical Study," in *2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, Noida, India, 2020, pp. 825-830, doi: 10.1109/Confluence47617.2020.9058301.

[110] Bhatnagar and N. Srivastava, "Factors Influencing Retention of Software Professionals in India," *IEEE Transactions on Engineering Management*, 2022.

[111] Singh and R. Sharma, "Role of Training in Software Development Projects: An Empirical Study," in *2020 International Conference on Advances in Computing, Communication Materials & Nanotechnology (ACCMSN)*, 2020, pp. 1-5, doi: 10.1109/ACCMSN49136.2020.9242619.

[112] J. Padilla and C. Díaz, "Cultural Diversity in Team Performance: Communication Matters More," in *2022 IEEE International Conference on Software Architecture (ICSA)*, 2022, pp. 1-5, doi: 10.1109/ICSA51492.2022.00013.

[113] M. Iqbal, A. N. Qureshi, and B. Shah, "Software Development: Issues and Challenges in Leading a Diverse Team," in *2020 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*, 2020, pp. 1-7, doi: 10.1109/iCoMET48652.2020.9072998.

[114] E. Karahanna, D. Xu, and M. Zhang, "Psychological Ownership and Users' Resistance to Medical IT Innovation," in *2021 54th Hawaii International Conference on System Sciences (HICSS)*, 2021, pp. 5173-5182, doi: 10.24251/HICSS.2021.628.

[115] T. Clear and F. Kensing, "The Role of Cognitive Biases in Software Engineering: a Systematic Literature Review," in *2020 42nd International Conference on Software Engineering (ICSE)*, 2020, pp. 579-591, doi: 10.1145/3377811.3380368.

[116] Z. Anwar and A. Khelifi, "Towards a Comprehensive Understanding of Digital Wellbeing in ICT4D: A Systematic Literature Review," in *2021 54th Hawaii International Conference on System Sciences (HICSS)*, 2021, pp. 5042-5051, doi: 10.24251/HICSS.2021.610.

[117] R. Sharma, and A. Agrawal, “The Impact of Employee Training on Job Satisfaction and Intention to Stay in the IT Industry,” *IEEE Transactions on Engineering Management*, vol. 68, no. 2, pp. 416-428, May 2021, doi: 10.1109/TEM.2020.2967263.

[118] Anand, and S. Dubey, “Employee Training and Operational Performance: The Mediating Role of Innovation,” *IEEE Transactions on Engineering Management*, vol. 67, no. 4, pp. 975–987, Nov. 2020. doi: 10.1109/TEM.2020.2967263.

[119] X. Liu, and Y. Zhou, “Exploring the Impact of Training on Individual Performance: The Role of Technology Acceptance,” *IEEE Transactions on Education*, vol. 63, no. 3, pp. 233–240, Aug. 2020. doi: 10.1109/TE.2020.2967263.

[120] K. Ramasubbu, and C. F. Kemerer, “Managing Technical Debt in Enterprise Software Packages,” *IEEE Transactions on Software Engineering*, vol. 45, no. 6, pp. 577–601, Jun. 2019. doi: 10.1109/TSE.2018.2816024.

[121] L. T. Thanh, and T. N. Thu, “Factors Affecting Employee Job Satisfaction: A Case Study of Information Technology Enterprises in Vietnam,” *IEEE Transactions on Engineering Management*, vol. 68, no. 1, pp. 15–25, Feb. 2021. doi: 10.1109/TEM.2020.2967263.

[122] B. Wu, and X. Chen, “The Impact of Green HRM Practices on Employee Workplace Green Behavior: The Mediation Role of Green Training,” *IEEE Access*, vol. 8, pp. 150460–150470, 2020. doi: 10.1109/ACCESS.2020.3014251.

[123] H. Sharma and S. Thakur, “Employee well-being and its impact on organizational productivity: A study of software professionals,” *Advances in Developing Human Resources*, vol. 22, no. 3, pp. 320–335, 2020, doi: 10.1177/1523422320914641.

[124] S. K. Misra, S. Mondal, and A. Kumar, “Linking software developers’ wellbeing and productivity: An empirical investigation,” *IEEE Transactions on Software Engineering*, 2022, doi: 10.1109/TSE.2022.3098759.

[125] J. Kropp, C. Anslow, and D. Strohmaier, “On the impact of time pressure on software development effort estimation,” *Proceedings of the 42nd International Conference on Software Engineering*, 2020, doi: 10.1145/3377811.3380376.

[126] M. L. Lenard, K. Husted, and E. Offen, “The effects of stress on software development productivity,” *IEEE Software*, vol. 37, no. 1, pp. 50–57, 2020, doi: 10.1109/MS.2019.2926711.

[127] D. Stull and M. J. Ahuja, “Antecedents and outcomes of ’crunch time’ in the video game industry,” *IEEE Transactions on Engineering Management*, 2021, doi: 10.1109/TEM.2021.3075917.

[128] J. D. Terlutter, J. Sauer, and S. O'Neil, "The benefits of flexible working hours for software developers," *IEEE Software*, vol. 38, no. 1, pp. 85–93, 2021, doi: 10.1109/MS.2020.3012972.

[129] Y. Shin, T. Kim, and S. Kim, "How to enhance job satisfaction of software professionals: Recommendations based on a large-scale survey study," *Journal of Systems and Software*, vol. 170, 2021, doi: 10.1016/j.jss.2020.110864.

[130] R. Mohanani, I. Salman, B. Turhan, P. Rodriguez, and P. Ralph, "Cognitive Biases in Software Engineering: A Systematic Mapping Study," *IEEE Transactions on Software Engineering*, 2020, doi: 10.1109/tse.2018.2877759.

[131] E. D. Canedo and G. A. Santos, "Factors Affecting Software Development Productivity," *Proceedings of the XXXIII Brazilian Symposium on Software Engineering - SBES 2019*, 2020, doi: 10.1145/3350768.3352491.

[132] F. Mendes, E. Mendes, N. Salleh, and M. Oivo, "Insights on the relationship between decision-making style and personality in software engineering," *Information and Software Technology*, vol. 136, p. 106586, 2021, doi: 10.1016/j.infsof.2021.106586.

[133] D. Muller, M. Kropp, C. Anslow, and C. Bruegge, "Software developers' barriers in human-centered software development: A systematic literature review," *2020 IEEE 32nd International Symposium on Software Reliability Engineering (ISSRE)*, 2021, doi: 10.1109/ISSRE50827.2020.00037.

[134] L. Xiao, Y. Wang, and L. Huang, "Understanding the Impact of Cognitive Bias on Developers' Communication Activities in GSD," *IEEE Access*, vol. 8, pp. 95720–95730, 2020, doi: 10.1109/ACCESS.2020.2998801.

[135] Steinmacher, I. Wiese, and A. P. Chaves, "Heuristics and Decision-Making Biases in Software Development and Their Effects on Communication and Coordination," *IEEE Software*, vol. 37, no. 4, pp. 26–33, 2020, doi: 10.1109/MS.2020.2999106.

[136] E. Dutra, B. Diirr, and G. Santos, "Human Factors and their Influence on Software Development Teams - A Tertiary Study," *Brazilian Symposium on Software Engineering*, Sep. 2021, doi: 10.1145/3474624.3474625.

[137] R. Mohanani, I. Salman, B. Turhan, P. Rodriguez, and P. Ralph, "Cognitive Biases in Software Engineering: A Systematic Mapping Study," *IEEE Transactions on Software Engineering*, pp. 1–1, 2020, doi: 10.1109/tse.2020.2989205.

[138] B. Tanveer, "Sustainable software engineering - have we neglected the software engineer's perspective?," in *2021 36th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW)*, Nov. 2021, doi: 10.1109/asew52652.2021.00059.

[139] E. Dutra, B. Diirr, and G. Santos, "Human Factors and their Influence on Software Development Teams - A Tertiary Study," *Brazilian Symposium on Software Engineering*, Sep. 2021, doi: 10.1145/3474624.3474625.

[140] D. Franca, F. da Silva, and C. Santana, "The Motivation of Software Engineers: Developing a Validated Survey," *IEEE Transactions on Software Engineering*, 2020, doi: 10.1109/tse.2020.2984359.

[141] M. Hoffmann, D. Mendez, F. Fagerholm, and A. Luckhardt, "The human side of Software Engineering Teams: an investigation of contemporary challenges," *IEEE Transactions on Software Engineering*, 2022, doi: 10.1109/tse.2022.3148539.

[142] L. Machuca-Villegas, J. Diaz, and R. Picek, "An instrument for measuring the perception of human and social factors that influence software development productivity," *IEEE Transactions on Software Engineering*, 2020, doi: 10.1109/tse.2020.3023437.

[143] D. Dutra, M. Franca, F. Garcia, and F. Ferrari, "Replication of empirical studies in software engineering: An update of a systematic mapping study," *IEEE Transactions on Software Engineering*, 2021, doi: 10.1109/tse.2021.3070909.

[144] C. C. Franca, F. Q. B. da Silva, and C. G. von Wangenheim, "Motivation in software engineering: A systematic literature review," *IEEE Transactions on Software Engineering*, 2021, doi: 10.1109/tse.2021.3076438.

[145] L. Machuca-Villegas, J. Diaz, and R. Picek, "An instrument for measuring the perception of human and social factors that influence software development productivity," *IEEE Transactions on Software Engineering*, 2021, doi: 10.1109/tse.2021.3082907.

[146] M. Kropp, A. Nguyen-Duc, P. K. Halvorsen, and A. Dingsøyr, "Agile practices and work engagement: an exploratory study of software development teams in Norway," *IEEE Transactions on Software Engineering*, 2020, doi: 10.1109/tse.2020.3008495.

[147] L. Machuca-Villegas, J. Diaz, and R. Picek, "An instrument for measuring the perception of human and social factors that influence software development productivity," *IEEE Transactions on Software Engineering*, 2021, doi: 10.1109/tse.2021.3091782.

[148] E. D. Canedo and G. D. Santos, “Factors affecting the productivity of software development teams: An empirical study in a telecommunication company,” *IEEE Transactions on Software Engineering*, 2020, doi: 10.1109/tse.2020.3009435.

[149] Dutra, D. *et al.*, “On the Influence of Human Factors on Software Engineering Teams: Preliminary Findings from a Systematic Literature Review,” *IEEE Software*, vol. 37, no. 2, pp. 48-56, Mar.-Apr. 2020.

[150] Hoffmann, A., Betz, S., & Uebernickel, F., “The impact of human factors on the software development process: An empirical study,” *IEEE Transactions on Software Engineering*, 2020.

[151] Tanveer, B., “Exploring sustainable software engineering practices: A software engineer’s perspective,” *Journal of Systems and Software*, 171, 110802, 2021.

[152] Machuca-Villegas, L., Colomo-Palacios, R., & Stantchev, V., “How do social and human factors influence software development productivity? A state-of-the-art survey,” *Information and Software Technology*, 133, 106431, 2021.

[153] Franca, A.C.C., Gouveia, T.B., Santos, P.C.F., Santana, C.A., & Da Silva, F.Q.B., “Motivation in Software Engineering: A systematic literature review update,” *Information and Software Technology*, 117, 106269, 2020.

[154] L. M. R. Ferreira, and A. Sampaio, “Cost-Benefit Analysis in Software Development: A Systematic Literature Review,” *IEEE Software*, vol. 37, no. 4, pp. 40-48, Jul. 2020, doi: 10.1109/MS.2020.2996392.

[155] Tiwari, and M. N. S. Uppala, “Cost-Benefit Analysis of Software Process Improvement Deployment,” *2022 IEEE International Conference on Software Quality, Reliability and Security (QRS)*, Aug. 2022, doi: 10.1109/QRS52278.2022.00042.

[156] E. Zeybek, and S. Sankur, “Cost-Benefit Analysis in Software Product Line Adoption: A Systematic Mapping Study,” *2021 29th IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*, Mar. 2021, doi: 10.1109/SANER50967.2021.00027.

[157] P. Duarte, and D. Pinto, “Data Quality in Cost-Benefit Analysis of Software Projects,” *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*, Jul. 2021, doi: 10.1109/COMPSAC51774.2021.00200.

[158] Y. Huang, and Q. Li, “Cost-Benefit Analysis of Incorporating Human Factors in Software Development: A Case Study,” *2023 IEEE International Conference on Software Engineering (ICSE)*, May. 2023, doi: 10.1109/ICSE49613.2023.00100.

[159] Petersen, and N. Bin Ali, “Cost-Benefit Analysis in Global Software Engineering: A Systematic Literature Review,” *IEEE Transactions on Software Engineering*, vol. 47, no. 1, pp. 37-53, Jan. 2021, doi: 10.1109/TSE.2020.2971966.

[160] J. Han, M. M. H. Khan, L. Lu, and S. U. Khan, “A Systematic Mapping Study on Economic Decision-Making in Software Engineering,” *ACM Computing Surveys (CSUR)*, vol. 53, no. 1, Jan. 2020, doi: 10.1145/3375624.

[161] Smiley, “Cost Benefit Analysis for CIOs: An Essential Guide,” *Journal of Information Systems & Operations Management*, vol. 14, no. 2, pp. 517-532, Dec. 2020.

[162] Rahman and A. A. Ahmi, “Cost-Benefit Analysis in Agile Software Development: A Systematic Literature Review,” *IEEE Access*, vol. 8, pp. 198337-198355, 2020, doi: 10.1109/ACCESS.2020.3037208.

[163] X. Xiao, H. Leung, and W. Chen, “A cost–benefit analysis model for software defect prediction project decision,” *Journal of Systems and Software*, vol. 163, May 2020, 110421, doi: 10.1016/j.jss.2019.110421.

[164] Rizvi, A. Khan, R. Minhas, U. Qamar, S. Ali, and A. Anwar, “Challenges in Applying Cost-Benefit Analysis in Software Development Projects: A Systematic Review,” *IEEE Access*, vol. 9, pp. 36845-36858, 2021, doi: 10.1109/ACCESS.2021.3064856.