




Article

Expediting Time to Market: Evaluating the Effects of Change Control Board Performance in Emerging Markets

Sanaullah Irfan ¹, Jamshid Ali ², Imdadullah Hidayat-ur-Rehman ² , Muddasar Ghani Khwaja ^{3,*} ,
Joanna Rosak-Szyrocka ⁴  and Attila Kovacs ⁵

¹ Bahria Business School, Bahria University, Islamabad 44000, Pakistan

² Faculty of Business Administration, University of Tabuk, Tabuk 71491, Saudi Arabia

³ Department of Business, School of Justice, Security and Sustainability, Staffordshire University, Stoke-on-Trent ST4 2DF, UK

⁴ Department of Production Engineering and Safety, Faculty of Management, Czestochowa University of Technology, 42-200 Czestochowa, Poland; joanna.rosak-szyrocka@wz.pcz.pl

⁵ Faculty of Business, Communication and Tourism, Budapest Metropolitan University, 1148 Budapest, Hungary; a.kovacs@metropolitan.hu

* Correspondence: khawajamuddasar@gmail.com

Abstract: This study aims to assess how the performance of Change Control Boards (CCBs) influences the relationship between requirements uncertainty and the Time to Market (TTM) of software projects in emerging markets. We collected data through a structured questionnaire, conducting surveys in project-based IT organizations across various cities in Pakistan. This research adopts a quantitative approach, employing software project houses as the unit of analysis. We selected 38 software projects out of 50 using a multi-stage sampling method and analyzed the data using Smart PLS 4.0 and SPSS 23. The results reveal that requirements uncertainty has a significant impact on Time to Market (TTM) conditions. Our study concludes that the presence and effective performance of CCBs can substantially reduce TTM. Additionally, higher CCB performance can expedite TTM, particularly when requirements lack precision initially. The study underscores the profound influence of requirements uncertainty on TTM in software projects. It provides organizations with insights into the critical role of CCBs, the consequences of uncontrolled changes in the absence or ineffectiveness of CCBs, and how requirements uncertainty affects software project development and timelines. This research contributes to software management and product development processes by highlighting the importance of requirement engineering during the planning phase to address uncertainty issues and enhance CCB performance.

Keywords: sustainable emerging markets; Time to Market; Change Control Board; software configuration management; work in progress; requirement uncertainty



Citation: Irfan, S.; Ali, J.; Hidayat-ur-Rehman, I.; Khwaja, M.G.; Rosak-Szyrocka, J.; Kovacs, A. Expediting Time to Market: Evaluating the Effects of Change Control Board Performance in Emerging Markets. *Sustainability* **2023**, *15*, 16085. <https://doi.org/10.3390/su152216085>

Academic Editors: Byung Il Park and Min-jae Lee

Received: 14 August 2023

Revised: 28 September 2023

Accepted: 6 November 2023

Published: 18 November 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

With the expansion in both the size and complexity of the applications in large organizations, the significance of configuration management (CM) has augmented. According to Siegmund [1], the cost of outages and other issues increases with system size. Therefore, businesses need to engage with multiple vendors, collaborators, or suppliers on various iterations of the exact specifications or ideas. It has been ever more challenging to gain a clear picture of the state of a system at any stage in its life cycle due to the expansion of systems and the steady increase in the number of interfaces [2]. Many businesses use ineffective or out-of-date CM techniques due to modifications to the software development process and would substantially benefit from optimizing these practices [3].

CM is crucial for lowering risks, cutting costs, and avoiding delays in the maintenance of software and its development operations [4]. Thus, refining a CM procedure can save expenses while reducing the number of faults present when software comes into production

and avoid implementation failure. Many large companies that were the first to begin implementing their own IT systems now face a significant challenge because many of their systems were originally never designed to communicate with one another, or at least not to the extent required today. Instead, they were built using different technologies altogether [5–7]. Consequently, a patchwork of linked legacy systems must be maintained, patched, and upgraded at escalating expense. The resulting architecture is frequently ill-equipped to meet present demands [8]. Therefore, the legacy systems could either be substituted entirely or developed and retained, which would involve rising costs and a decreasing investment return. It is expensive and risky to replace a legacy system since there is the risk of losing crucial business data of the system [9].

A more up-to-date approach focuses on reforming the system or refining and reorganizing it so that it is simpler to maintain and update and better fulfills today's demands. This reengineering method is usually a cost-effective intermediate approach between these two limits. These reengineering processes are called business transformation [10]. Consolidating systems and moving related items into a single system could be a vital constituent of transformation projects. Customer Relationship Management (CRM) software often plays a significant role in these transformation efforts; meanwhile, it is becoming ever more essential for it to provide a complete image of a customer [9]. Large CRM systems, which often manage millions of customers, are crucial sources for customer care and the basis of any web-based self-help facilities when a business deals with its customers. Because of poor publicity and high prices in the event of interruption or severe production failure, reducing the number of faults that reach manufacturing in such a system is crucial. Transformation projects often have significant delays, increasing prices, and poor features due to their complexity, leading to many problems going into production [10]. Because of a lack of internal involvement or a decision to allow someone else to take the risk, many major developments are contracted out, possibly contributing to the problem. In many cases, outsourcing may limit the company's control of the business [8,11].

CM offers a unique chance to recover command over this chaotic and expensive situation. The fact that CM is the mechanism that communicates with software development, receives modifications, and eventually handles deployments makes it a strong target for minimizing issues and cutting costs [12]. Because everything that enters production must pass through the CM cycle, even if the business does not directly manage the software development life cycle, quality control standards could be applied at various stages to identify issues as soon as possible and eventually stop them from being delivered. As a result, a business may, as a first step, assess its CM process to optimize it and uncover a wider variety of problems earlier.

Rapid changes are expected during software project implementation, which may impact the projects from various perspectives including cost, additional complexities, and delays in Time to Market. To assess these changes, a board is formed called the Change Control Board (CCB), which is a group of people tasked with evaluating or approving modifications to baselined work. If these changes that come to the CCB are not managed effectively, they may impact software projects by increasing the work in progress and delays in Time to Market. However, there are a number of factors which also affect the performance of the CCB and may affect the Time to Market. Firstly, the composition of the CCB plays a pivotal role, with the expertise and experience of its members significantly impacting decision-making. Clear and efficient decision-making processes are equally essential, requiring well-defined procedures and transparent criteria for evaluating and approving change requests [13]. Effective communication and collaboration among CCB members, project teams, and stakeholders are vital, ensuring that all perspectives are considered. Adequate resource allocation, encompassing both time and personnel, is crucial to prevent overburdening and delays. Furthermore, the quality of change request documentation and prioritization, as well as alignment with project goals, all contribute to CCB performance [14]. Lastly, ongoing monitoring and feedback mechanisms help evaluate the effectiveness of decisions and enable continuous improvement in managing changes

within software projects [15]. Rapid Time to Market is essential for the competitive success of projects as it ensures a faster breakeven point, an edge over competitors, longer market life, and a higher return on investment. Software development companies, especially startups with limited resources, may not have an effective CCB as part of their system [16].

Therefore, this research focuses on determining the impact and role of the Change Control Board performance on the Time to Market of software projects regarding requirements uncertainty. This research is backed by theory W, which focuses on the importance of Time to Market software projects and creating a win-win situation for all stakeholders. To address the objectives of the study, the paper consists of an in-depth literature review, followed by empirical analysis, discussion, conclusion, and recommendations.

In the software industry, the ability to deliver products to the market quickly has become a critical factor for success. Time to market (TTM) refers to the duration between the initiation of a project and the release of the final product. Several factors can influence TTM; one such factor is the level of requirement uncertainty. Requirement uncertainty refers to the ambiguity and variability in the understanding and definition of project requirements. It has been observed that higher levels of requirement uncertainty often lead to delays and inefficiencies in software projects, consequently impacting TTM. This research seeks to provide empirical evidence on how requirement uncertainty affects TTM in the software industry and whether the performance of the CCB can moderate this impact. By investigating these relationships, this study aims to shed light on the significance of effective requirement management and the role of the CCB in mitigating the challenges posed by requirement uncertainty.

This study addresses a gap in the project management literature by emphasizing the impact of requirement uncertainty on TTM. It highlights the importance of recognizing uncontrolled and untimely requirements as factors influencing TTM in software projects. By addressing this area, the research provides empirical evidence and supports understanding of this ground reality. Additionally, this study emphasizes the significance of precise and accurate requirement gathering and the development of a work breakdown structure based on customer-provided requirements. While agile methodologies are prevalent in the IT industry, this research highlights the need to not overlook the process of requirements gathering and scope definition. Engaging with customers extensively to clarify requirements and elaborate work packages before development starts can reduce uncertainty and improve project performance, ultimately reducing TTM.

The research is of substantial importance as it emphasizes the importance for businesses in developing economies to adopt sustainable practices. It establishes a clear link between the performance of Change Control Boards and sustainability, highlighting that improved change management processes can lead to more sustainable practices. Our study is guided by the idea that better Change Control Board performance leads to reduced resource usage, less waste generation, and minimized environmental impact, all in line with sustainable development goals. Through a strong methodology and empirical evidence, the research shows how more effective change control processes contribute to better sustainability, making businesses in emerging markets more environmentally conscious and economically effective. Additionally, the findings stress the importance of integrating sustainable practices into change management strategies, demonstrating the potential for long-term cost savings and improved reputation. This work not only enhances the understanding of strategies relevant to Time to Market but also offers practical insights for businesses and policymakers aiming to promote sustainability in emerging markets.

For the industry, the findings have practical implications for project management in the IT industry. Organizations can benefit from a clear understanding of the impact of requirement uncertainty. By training project teams to effectively engage customers and manage the requirements-gathering process, they can deliver the desired results without delays. This helps retain existing customers and attract new ones, leading to a prominent position in the market. This study caters to established firms as well as new startups, as customer satisfaction is crucial for every business.

Furthermore, the research highlights the moderating role of CCB performance in situations of requirement uncertainty. It suggests that project managers and leaders should focus on the requirement gathering and engineering phase to minimize uncertainty and deliver products to the market on time. The study recommends exploring other organizational or contextual factors related to product uncertainty and encourages further research to improve the model by considering additional mediators and moderators such as the role of product owners, quality assurance, and customer engagement.

2. Theoretical Background

2.1. Software Development Methodology

Software is a group of computer programs, instructions, and documentation that give a computer system the ability to accomplish tasks. Software is developed and produced in projects where groups of talented individuals collaborate to produce software components that will operate according to the software product's guidelines [17]. The area in which these initiatives are developed is known as the Software Development Process (SDP), which is characterized as an organized series of tasks that, when finished, produce a software product given to a client [17]. These projects require much information, which means that when creating a software product, knowledge applied to every component is the most valuable commodity. The increasing crystallization of information into a language that a machine can understand and operate, according to Robillard [18], is what constitutes software development. The individuals participating in the product creation are the basis of the knowledge employed during a software project. As a result, the software development process requires managing not only the technical problems that arise during the creation of a piece of software but also the method by which individuals carry out their tasks.

This fact has influenced the development of new methods for tackling the problem of software development project management. These methods aim to build a structured SDP environment with regulated programming, design, and component analysis for software products. These methods are categorized under a brand new branch of science called software engineering that addresses problems with software development. Monitoring, costing, scheduling, modeling, analyzing, specifying, developing, executing, evaluating, and sustaining software are tasks that fall under the purview of software engineering [19].

To efficiently plan and carry out these phases, the software development model divides software development into separate stages. Other approaches are available, including agile, spiral innovation, waterfall, and modeling. Alternative approaches may be used, reflecting the organization and system's needs. It is usual for firms to develop their own processes, either a variation of an existing implementation of a technique or a combination of various processes [20]. A single process may frequently be inadequate to suit a company's unique requirements. Waterfall, prototypes [14], spiral design, and other agile techniques are some of the most popular methodologies.

Conventional techniques, like the waterfall approach, have separate phases carried out in order. On the opposite end of the spectrum, we have more recent agile approaches built on an iterative process where many phases can be carried out concurrently [21]. According to Hoare [22], errors are more likely to occur during design and specification operations. The earlier in the process, the more costly the correction will be. The manufacturing sector is the origin of sequential approaches because design changes made late in the development cycle are costly. When people initially tried to give software design more structure, these methodologies were the only ones available, and ever since, sequential approaches have come under fire for being inefficient for software development. The Figure 1. presents outline of a sequential software development process.

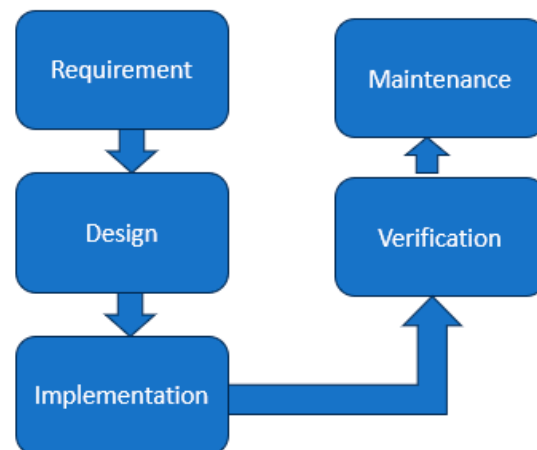


Figure 1. Outline of a sequential software development process.

The software development project unfolds through a series of crucial phases, each serving a distinct purpose. It commences with the “requirement” phase, when project teams engage with stakeholders to gather and define the system’s functionalities and user needs. Subsequently, in the “design” phase, comprehensive blueprints are created, outlining the software’s architecture, structure, and user interfaces. The “implementation” phase brings these designs to life through coding and programming, transforming abstract concepts into functioning software. Following implementation, the “verification” phase rigorously tests the software for defects, ensuring it aligns with the specified requirements and quality standards. Finally, in the “maintenance” phase, the software enters its operational stage, where ongoing updates, bug fixes, and enhancements are performed to ensure its longevity and adaptability to changing user needs and technological landscapes. These phases collectively form the software development lifecycle, guiding projects from concept to deployment and beyond, while maintaining focus on quality, functionality, and stakeholder satisfaction.

Sequential techniques attempt to address the issue of late requirements and design changes. Writing practical user requirements and design documentation that genuinely reflect what the end users desire has, sadly, frequently proven challenging. By employing an iterative approach with continuous integration, agile methods have concentrated on resolving the issue of capturing and designing documentation. This allows reviews from stakeholders to be gathered at a preliminary phase when it is feasible to make significant changes without substantially raising costs [10].

The quantity and complexity of the methodology’s rules and practices serve as the basis for classifying software development approaches into heavyweight and lightweight methods. While lightweight techniques concentrate on quick multiple iterations and rely on team expertise instead of comprehensive documentation, heavyweight approaches emphasize detailed evidence and inclusive design. Most agile techniques are lightweight, while all older development methodologies tend to be heavyweight [13]. The acceptance of change is the primary distinction between these two categories of techniques. Traditional heavyweight approaches prohibit additional revisions after the specifications have been frozen at an early stage. Agile techniques, in contrast, permit continuous change based on input from prior iterations. The capacity to manage change well typically determines whether a project succeeds or fails, and the notion that change is inescapable is one of the main motivations of agile techniques [5].

The software configuration method is improved by adapting to the chosen software development approach, considering the significant variations in how modifications are handled and how crucial system maintenance is in implementing change. While lightweight techniques with numerous modifications benefit more from a procedure that emphasizes certainty that adjustments are supported and completed in a controlled way, heavyweight methodologies, with a concentration on complete requirements from the beginning and

a few or no modifications, benefit more from a method that emphasizes order, ensuring that documentation is completed and evaluated correctly. This implies that the software development approach employed also influences the best practices for the CM that is adopted [4,9].

The decision was made to concentrate on optimizing software configuration procedures integrated with complex, waterfall-style software development models because it is practically impossible to create a technique for software maintenance performance improvement that is impartial to the software development model. This is because legacy systems typically require more optimization and typically employ heavyweight approaches. Hence, it is likely that this optimization technique will be used the most [1,14].

2.2. Software Development Best Practices

Processes for managing software configuration can be entirely or partially automated with version control tools or be carried out physically.

2.2.1. Version Control Software

The administration of modifications to a collection of items, most frequently source code and documentation maintained in a repository, is basically performed using multiple versions, also called source control or revision control. A new version of an item is created with each modification. Each new version is identifiable by a sequence number, often a counter with a starting value of 0 or 1 that increases by 1 with each recent update. Version numbers can be local or global, meaning they may only be attached to the item that has changed or to every item in the repository [20].

Every modification has a timestamp and a user assigned to it. Most version control programs keep track of modifications on a file-by-file basis, which makes sense but can lead to issues when splitting, combining, or replacing files. While less intuitive, more recent version management systems, analyze changes to the data all at once [14]. This makes operations like merging, separating, and updating files considerably simpler. Graph theory can be used to conceptualize revisions or modifications to an item, and many graphical version control solutions opt to depict upgrades or modifications to an item in this way [23].

2.2.2. Requirements and Documentation

How thoroughly each need has been gathered and documented is a crucial factor determining the success of any software project, as these records will form a significant portion of future baselines. Due to this, version management must also be applied to requirement specifications, project schedules, SCM proposals, and other documents [24]. To keep the documentation current, it is just as crucial to developing new documentation as it is to maintain existing material. Maintaining good documentation instead of keeping information throughout the organization can reduce serious hazards such as the “Truck or Bus factor” [25].

2.2.3. Communication

Software engineering frequently involves big teams working on various system components while occasionally dispersed across several time zones and nations. Even different vendors or suppliers may design various modules or components [26]. A higher risk of misunderstandings and communication failure exists when people who do not know one another are from various cultures and when the opportunity for face-to-face interaction, such as meetings and conversations, is constrained by distance, both physically and temporally. Software systems are also naturally complicated and frequently abstract, thus raising the possibility of communication issues even more [27].

2.2.4. Shared Data, Multiple Maintenances, and the Simultaneous Update Problem

Shared data is one of the first issues that might arise when software development grows, programs become more extensive, and teams (instead of individuals) interact using

the same resources, such as several people utilizing and modifying variables for an App-Store in a database [13,26]. In any setting where several separate programs, programmers, or groups use the same resource, such as a standard requirement or a database table, information sharing and resources can develop into a severe issue. The shared resource could stop working altogether or function poorly the following time if one or more parties are unaware of changes made by other parties [5].

It can also be challenging to debug software if it suddenly stops operating with no noticeable change. In software design, this issue is typically resolved on various levels by giving developers and teams access to their supplies, i.e., several versions of the same resources are produced. However, this does not resolve every issue; instead, separate, multiple maintenance issues arise [27].

The term “multiple maintenance problems” relates to the challenge of keeping track of many variants of a shared resource, such as the number of copies and what modifications have been made to which copies. The copies of the service must usually all be identical. The simultaneous update issue can be caused by employing a repository, or centralized network storage, to resolve this issue [28]. The core issue with simultaneous updating is that there is no method to regulate changes, i.e., changes to a resource. Therefore, two parties may accidentally overwrite each other’s modifications, thereby losing, for instance, a bug fix.

All these issues can be resolved by adequately implementing SCM using practically any current version control system. Access control can be applied in a repository where the source code is/can be stored. A modification request must be submitted before any changes may be made, and the resource cannot be accessed until the request has been accepted [29]. Before the modification is incorporated into the repository, it will be executed, examined, and approved. Controlled updates resolve the shared data problem, and the simultaneous update and multiple maintenance issues are resolved by limiting the number of repositories and users who can simultaneously check out configuration items [26].

2.3. Work in Progress

Work in progress, also known as work in process, is the work taken up by the developers but not yet converted to a milestone or added to the final product [15]. According to PMI, it is something that has started but is not yet completed, and one of the reasons for increased WIP is constantly incorporating new changes and requirements to the ongoing tasks. Sjøberg [30] suggested that placing limits on work in progress helps in a continuous flow of tasks and helps in reducing Time to Market. He also concluded that work in progress should be balanced to increase productivity and lead time.

2.4. Uncertainty

When coping with project modifications, uncertainty is regarded as one of the most crucial and difficult challenges. Logically, uncertainty is the state of knowledge in which every choice leads to a group of consequences but when the likelihood of each outcome is unknown for the decision-making process [31]. In the early stages of IT initiatives that take place in a complex environment, uncertainty is acknowledged as one of the most significant and inescapable issues. Uncertainty is regarded by Hameed [28] as being extremely invulnerable to examination; it is the prospective risk that is left in the background when every threat has been identified [31]. Uncertainty indicates a possible risk, but we are unsure of what form it will take. According to Standish research from 2015, just 39% of projects are successful, meaning they are finished on time and within the allotted budget; 43% of projects go over budget, schedule, and specifications; and 18% are abandoned or never started [31].

2.5. The Role of CCB

The Change Control Board (CCB) plays a pivotal role in software development projects by serving as the governing body responsible for evaluating, approving, and managing

changes to the project's scope, requirements, and overall direction. Its primary purpose is to maintain control and stability in the project by carefully assessing proposed changes to ensure alignment with project objectives, timelines, and resources [14]. The CCB acts as a gatekeeper, ensuring that changes are thoroughly analyzed for potential risks, impacts, and dependencies before being approved or rejected. Moreover, the CCB contributes to maintaining project integrity by preventing unauthorized or unmanaged changes that could disrupt development processes. It fosters effective communication and collaboration among project stakeholders, facilitates risk mitigation, and enhances project transparency [30,32]. Ultimately, the CCB's role is critical in ensuring that software development projects stay on track, adhere to established goals, and deliver high-quality products within specified timelines and budgets.

2.6. Research Hypothesis

2.6.1. Requirements Uncertainty (RU) and Time to Market (TTM)

The degree to which requirements change during the project lifecycle is known as requirements uncertainty [32]. Early requirements gathering and architectural decisions lead to a successful software product [9]; therefore, effective management is required to deal with requirements changes, and if these changes are not managed properly, the result may be the project's failure and loss of business [33]. New requirements and changes may necessitate architectural changes in the software project, and a team may spend much of its time dealing with these changes, thereby increasing work in progress without adding value to the system [28].

Khwaja et al., [3] states that the time between the conception of a product and its release to the market is described as a Time to Market (also known as TTM or time-to-market). These authors argue that TTM is critically important to maintaining a strategic edge. By boosting TTM, a company may take advantage of market opportunities and beat its direct competitors to the market. A company can gain market share by streamlining development, change management, and marketing processes to reduce Time to Market (also known as the "first-mover advantage" [24]. If a firm is the first to market, the product introduction has a more substantial effect, resulting in higher sales and profit margins than slower competitors. Apart from assessing the software system's defect levels, another essential characteristic of the software product is its market release schedule [34]. Busari [9] stated that requirements uncertainty adds complexities to the project in terms of design and development, resulting in more work, and may impact the Time to Market of a project. Based on the above literature and the careful analysis of the already available literature, this study proposes the following hypothesis:

H1: *Requirements uncertainty significantly increases the Time to Market of software projects.*

2.6.2. Change and Configuration Control and CCB Performance (CCBP)

According to IEEE Std-610-1990, configuration control is a CM component and entails the formal designation of configuration management, review, collaboration, approval or rejection, and execution of changes to those items. Configuration control is the mechanism that collectively enables all modification processes involving configuration items. The mechanisms used in configuration control, probably the most critical component of configuration management, enable changes, but only permitted modifications can be performed systematically. The procedures may vary significantly from one configuration item to the next; for instance, source code may call for different checks and testing than documentation. As configuration control is at the core of software CM(SCM), it is likely to be a significant factor in any implementation of SCM. Because these procedures are always employed repeatedly, this is perhaps where modifications can have the most significant influence.

The client typically initiates changes, and these might have a variety of causes, such as altered needs or end-user feedback. Proposed changes and problems may occasionally be handled independently, but the procedure is usually the same. A change request (CR) must

be categorized for the Change Control Board (CCB) to process it more effectively. Depending on the company's requirements, many classification techniques may be employed, but some of the most popular include severity, significance, effect, and price. The CR author or configuration manager (CM) team members perform the first categorization, but the CCB often makes the final determination [35]. All modifications must be assessed in terms of impact, price, modification to schedule, etc., for the CCB to make wise decisions.

Different persons might carry out different aspects of the assessment; for instance, a developer may predict how long it will take to make a change, a test manager may predict how long it will take to test, etc. Pre-evaluations and cost–benefit analyses are frequently included in the analysis stage to avoid spending time on change requests that will not be executed due to financial or logistical constraints. The CCB may decide after the analysis is completed. The actual procedure for handling approved and refused CRs can differ, but generally, refused CRs are returned to the original author with an explanation, and the modifications that have been authorized are transmitted to the design team to be put into practice. The design team can implement a modification after it has been accepted, but it must first be checked. Depending on which stage of the development cycle the CR is implemented in, the precise procedure and technique for verification may change. The test and validation methods are usually more rigorous the later a modification is made. The changed configuration identifications (Cis) can now be checked in, and a new benchmark is produced if the verification is successful. To reduce overheads, changes are rarely developed, verified, and distributed in isolation; several changes are implemented, evaluated, and deployed at the same time.

Planning and managing requirements change in software development is a challenging task. A CCB is a committee or group of experts like software engineers, quality assurance experts, and managers who decide whether to implement the requested project changes or not [3]. The CCB is part of an integrated change control system responsible for accepting or rejecting changes in scope and coordinating these changes among stakeholders to stop ever-expanding expectations, which is called scope creep [36]. According to Fahmy et al., [37], documentation plays a significant role in enhancing the CCB's performance compared with other tools. They also stated that better CCB performance benefits the management of new and existing projects. Hence, based on the above-cited literature and careful analysis of the already available literature, this study proposes the following hypothesis:

H2a: Higher CCB performance significantly decreases the Time to Market of software projects.

H2b: CCB performance moderates the relationship between requirements uncertainty and Time to Market of software projects.

2.7. Theoretical Framework

The Theoretical Framework is presented in Figure 2 below:

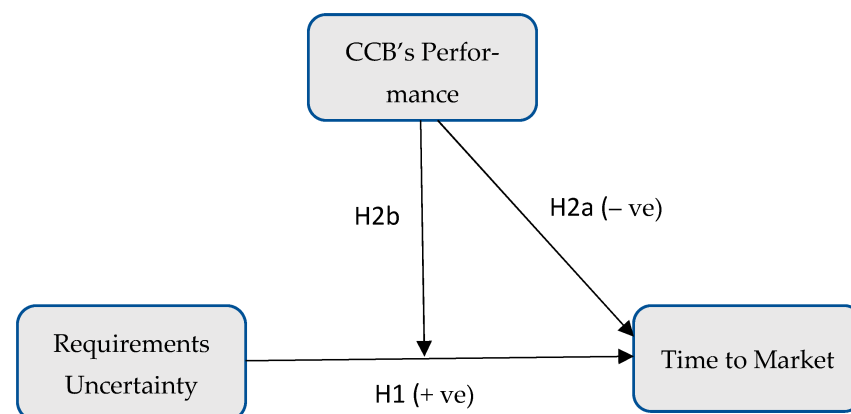


Figure 2. Framework for study.

3. Research Methodology

This study was quantitative in nature. The data were collected using an adaptive questionnaire from previous studies. The first portion of the questionnaire collected demographic information from the respondents, while the second portion was used to collect information about the constructs used in the study. For the second part, closed-ended questions were used to obtain precise answers and measure the items. The survey was based on a five-point Likert scale.

Data Collection

Quantitative data were collected from Software company managers and executives. The study aimed to obtain data from senior professionals. The target population was different software development companies operating across Pakistan that were registered with the Pakistan Software Export Board (PSEB). The head offices of these companies are located in the major cities such as Lahore, Karachi, Islamabad, and Rawalpindi.

Moreover, a multi-stage sampling technique was applied in the study. In the first phase, major/large software houses were clustered in four groups, so that all regions had equal opportunity to be included in the study. Similarly, through quota sampling, only project managers and senior members of the organizations who had market, development, and customer management experience were included in the study. In the third phase, a simple random sample was used to collect data from respondents. The data were collected through a self-administered questionnaire. The unit of analysis for the data was organization; therefore, when more than one respondent from an individual organization participated in the study, their weighted mean was included in the data. After data collection, the data were screened for anomalies and outliers. However, the data collection was self-administered; therefore, no questionnaire was discarded from the study. Finally, a total of 38 organizations participated in the study.

4. Findings and Results

Smart-PLS 4.0 and SPSS 23 were used in this study to analyze the collected data. A two-step approach was applied. The first step was to evaluate the measurement model for validity and reliability, and the second involved assessing the structural model for hypothesis testing [38]. The descriptive statistics were generated using SPSS V29.

4.1. Demographics of the Study

The demographics of the study are presented in Table 1. The data were collected from both genders, with 76.3% of participants being male and 23.7% female. Similarly, regarding the qualifications of the respondents, the majority of respondents possessed bachelor's degrees i.e., 65.8%, and 31.6% of the respondents had master's degree qualifications, while 2.6% had Ph.D. qualifications. Furthermore, during data collection, more attention was paid to collecting data from experienced professionals with at least 5 years of professional experience. The data show that most respondents (68%) had 5 to 10 years of experience, and 21.6% of respondents had 11 to 20 years of work experience. Furthermore, the frequencies of the job designations indicate that the data were collected from professionals with high levels of experience and subject knowledge (Table 1).

Table 1. Demographics of the Study.

Gender	Male	29	76.3%
	Female	9	23.7%
Qualification	Matric	0	0
	Bachelors	25	65.8%
	Masters	12	31.6%
	PhD	1	2.6%

Table 1. *Cont.*

Work Experience	5–10 years	26	68.4%
	11–20 years	12	21.6%
	21–30 years	0	0
	30+ years	0	0
Designation	Project Manager	10	26.31%
	PMO	2	5.2%
	Software Engineer	14	36.84%
	Human Resource	1	2.6%
	Quality Assurance	6	15.7%
	Technical Lead	5	13.15%

4.2. Descriptive Statistics

The data collected in this study are presented as descriptive statistics in a tabular format (Table 2). The number of responses is highlighted along with the mean, standard deviation, minimum and maximum values. The mean represents the variable's mean, and the standard deviation shows how much the values deviate from the mean [39]. All variables in this study (i.e., requirements uncertainty, CCB performance, and Time to Market) were analyzed using a 5-point Likert scale, so scores range from 1 to 5 (minimum = 1 and maximum = 5). Table in Section 4.3.4 shows the total number of respondents and the corresponding values.

Table 2. Descriptive statistics.

Variables	N	Min	Max	Mean	Standard Deviation
RU	38	1	5	3.83	0.32
CCBP	38	1	5	3.87	0.53
TTM	38	1	5	3.54	0.60

4.3. Measurement Model Assessment

The measurement models were evaluated by calculating reliability and validity. For reliability testing, first internal consistency reliability was assessed, followed by the indicators for reliability and composite reliability. To assess the validity, convergence and discriminant validity were examined.

4.3.1. Reliability and Convergent Validity Analysis of the Scale

Reliability is confirmed by repeatedly testing an item on a scale and obtaining consistent results. Cronbach's alpha (>0.7) was examined to assess internal consistency reliability. The threshold value used for composite reliability and indicators' reliability was 0.7 [40]. The results in Table 3 reveal that all values of Cronbach's alpha and composite reliability are above 0.7, showing that the scales are reliable. According to the recommendations of Turi et al. [41], indicators (namely, CCBP3, CCBP6, RU1, RU4, RU7, RU9, RU10, and RU12) having loadings of less than 0.7 were removed from further analysis. The convergence validity is checked using AVE and is confirmed if the sampled mean-variance value is more than 0.5 [3]. In this study, all variables have AVEs above 0.5, indicating convergent validity.

Table 3. Reliability and convergent validity analysis.

Variables	Items	Factors	Cronbach's Alpha (CA)	Composite Reliability (CR)	Average Variance Extracted (AVE)
Change Control Board Performance	CCBP1	0.851	0.899	0.917	0.712
	CCBP2	0.867			
	CCBP3	0.019			
	CCBP4	0.870			
	CCBP5	0.781			
	CCBP6	0.013			
	CCBP7	0.834			
Requirement Uncertainty	RU1	0.516	0.901	0.912	0.672
	RU2	0.844			
	RU3	0.720			
	RU4	0.386			
	RU5	0.856			
	RU6	0.747			
	RU7	0.563			
	RU8	0.758			
	RU9	0.653			
	RU10	0.537			
	RU11	0.828			
	RU12	0.085			
Time to Market	TTM1	0.855	0.901	0.907	0.717
	TTM2	0.813			
	TTM3	0.843			
	TTM4	0.826			
	TTM5	0.894			

4.3.2. Discriminant Validity Analysis of the Scale

The first criterion for checking the validity of the discriminant is called Fornell–Lacker, as shown in Table 4. According to this method, the variable's value concerning itself must be higher than the value of variables according to other structures [41]. That is, the value of the square root of AVE for a variable must be significant compared with its correlation with the other corresponding variables [42]. This indicates that the variables should exhibit higher variance than their related variables. Table 4 shows the values of all variables and the square root of AVE for this study. The diagonal values show the correlations between the latent variables, and the diagonal values in the table represent the square root of AVE.

Table 4. Discriminant validity (Fornell–Larcker).

Variables	Requirements Uncertainty	Change Control Board Performance	Time to Market
Change Control Board Performance	0.844		
Requirements Uncertainty	−0.536	0.820	
Time to Market	−0.618	0.609	0.847

In Table 5, the variables show the most significant variation relative to themselves, ensuring the validity of the discrimination. For discriminant validity, the Heterotrait–Monotrait Ratio (HTMT) method was proposed [7]. This method is used to check the discriminant validity between variables. The threshold for HTMT is 0.9, and all values must be less than 0.9 for the data to be valid [43]. As shown in Table 5, pairwise comparisons confirmed the validity of the discrimination, with all values below 0.9 [7].

Table 5. Discriminant validity (HTMT).

Variables	Requirements Uncertainty	CCB Performance	TTM
Change Control Board Performance			
Requirements Uncertainty	0.580		
Time to Market	0.652	0.665	

4.3.3. Structural Model Assessment

The coefficient of determination R2 was evaluated by employing the PLS algorithm. The R2 is 0.513, which indicates that the model of the study has 51.3% explanatory capability in describing the Time to Market of software projects in terms of Change Control Board performance and requirements uncertainty. The R2 value 0.513 is exhibiting a moderate level of explanatory power.

For testing of hypotheses, the bootstrapping technique was employed using 5000 sub-samples. Default settings were adopted in terms of the remaining settings of bootstrapping. To gauge our outcomes, the study examined the path coefficients with corresponding t and p estimates. Figure 3 below demonstrates the path coefficients with relevant p and t values. The outcomes of structural model analysis provided support for all of our hypotheses. Our hypothesis H1 is relevant to the significant impacts of RU on TTM. RU has a significant positive relationship with TTM ($\beta: 0.377, p = 0.016$). Thus, H1 is supported. The CCBP has a negative significant relationship with TTM ($\beta: 0.40, p = 0.003$), indicating that higher performance of the Change Control Board significantly decreases Time to Market. This finding supports hypothesis H2a. Our next hypothesis, H2b, regards the moderating effects of CCBP on the RU→TTM relationship. The moderating effects on this relationship are significant, which provides support for hypothesis H2b. The empirical outcomes provide support for all of our hypotheses, and thus, the model of the study is validated.

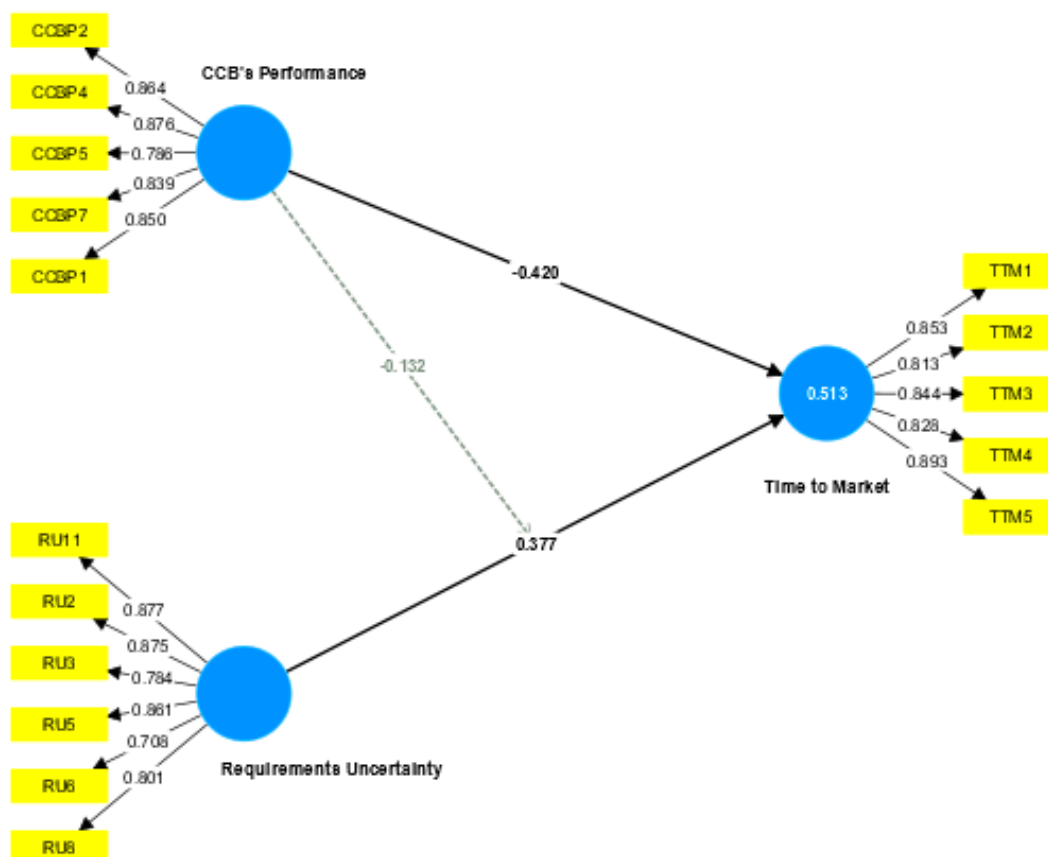


Figure 3. Measurement model graphical output.

4.3.4. Moderating Effects of CCBP on RU→TTM Relationship

This study examines the moderating effects (H2b) of CCPB on the RU→TTM relationship. The results in Table 6 reveal that the interaction of CCBP and RU on TTM is significant ($\beta: -0.132, p = 0.044$), which demonstrates that the RU → TTM relationship is reliant on CCBP. The direct effect of RU on TTM is $\beta: 0.377$. This result indicates that the intensity of RU → TTM is 0.377 when there is an average degree of CCBP. For increased levels of TTM (i.e., if TTM is improved equal to 1 standard deviation unit), the RU and TTM correlation is decreased by the value of the communication term (i.e., $0.377 - 0.132 = 0.245$). Enhanced levels of CCBP will lead to a weaker relationship between the RU and CCBP and vice versa. Figure 4 presents structural model outcomes. Figure 5 below shows a graphical representation of the slope analysis. The colorful lines in Figure 5 demonstrate moderating effect outcomes.

Table 6. Structural model assessment.

	Path	Path Coefficient	Standard Deviation	T Stats	p Values	p-Value
H1	RU → TTM	0.377	0.156	2.409	0.016	**
H2a	CCBP → TTM	-0.420	0.143	2.940	0.003	***
H2b	CCBP × RU → TTM	-0.132	0.066	2.011	0.044	**

Note: ** $p < 0.05$; *** $p < 0.01$; NS = Not Significant.

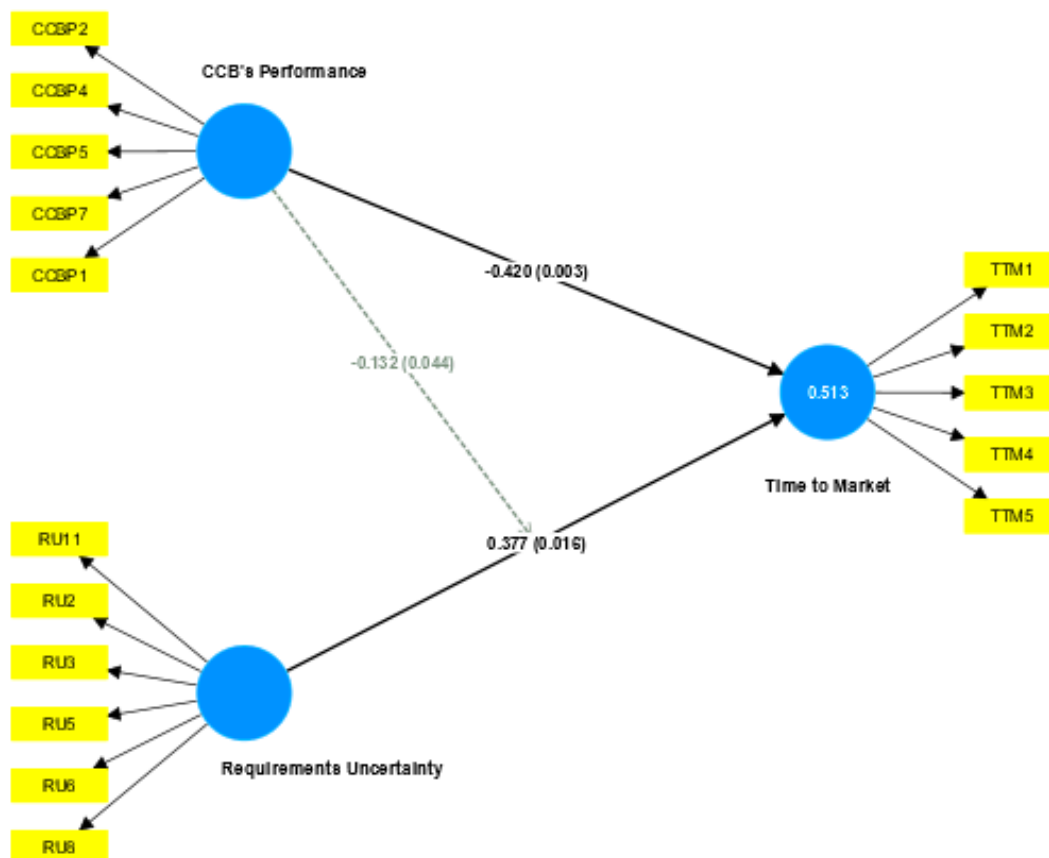


Figure 4. Structural model with inner model t-values from bootstrapping analysis.

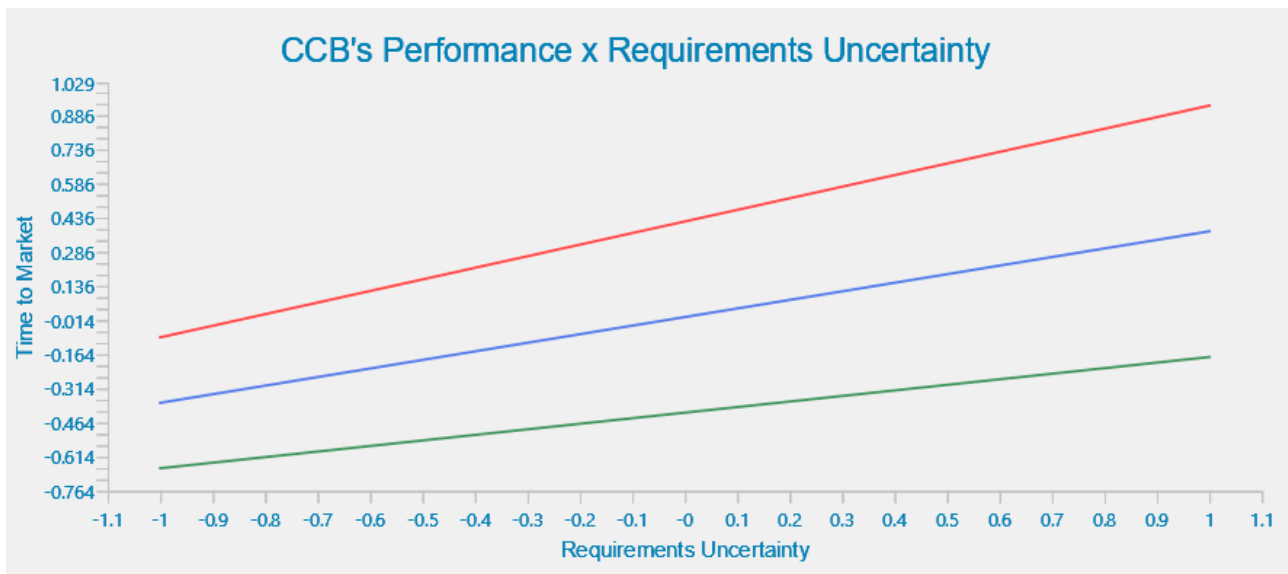


Figure 5. Slope analysis (moderating effects) of CCBP \times RU \rightarrow TTM.

5. Discussion

The core agenda of this research was to study the relationship between requirements uncertainty and Time to Market of software projects with CCB performance as a moderator. The impact of requirements uncertainty on Time to Market is researched, and along with it, the moderating effect of CCB performance is investigated. For this purpose, data were collected from organizations with IT-based projects in multiple cities in Pakistan [44]. According to the study results, requirements uncertainty significantly impacts TTM conditions in the software industry. The results are in line with many previous studies [6,45], and contextually also, it is certain that the greater the uncertainty in the market, the more uncertain will be the product delivery. Therefore, stable logistics support is required for the products to be in the market with the customers.

The second focus of the study was to assess the moderating impact of CCB performance on the requirement uncertainty and TTM of the software product. The empirical results provide support for our hypothesis. According to the results, higher performance of the Change Control Board can lead to a significant decrease in the time taken for delivery of the software project to market. Therefore, this study recommends practitioners to concentrate on improving Change Control Board performance if the in-time delivery of projects is difficult due to requirements uncertainty. Improving performance can be helpful in the prevailing circumstances. According to Malik et al. [46], change management becomes very complex due to many dependencies, and requirements uncertainty is one of them.

The uniqueness and novelty of this research study lie in its comprehensive exploration of the intricate relationship between requirements uncertainty, Time to Market (TTM) in software projects, and the moderating role of Change Control Board (CCB) performance. While agile methodologies have been the focal point of much research, this study delves deeper to shed light on the often-overlooked significance of effectively managing and controlling requirements in reducing TTM. By providing empirical evidence of the substantial impact of requirement uncertainty on project performance and TTM, this study fills a critical gap in the project management literature. Additionally, its findings highlight the practical importance of engaging customers extensively, clarifying requirements, and adopting agile strategies that prioritize thorough requirements gathering within the IT industry. The innovative aspect of this research is further exemplified by its emphasis on the role of technology-driven tools, particularly those based on artificial intelligence, in enhancing CCB performance, an area that has received limited attention in the existing literature. In

essence, this study not only advances theoretical understanding but also offers practical insights, making it a valuable contribution to both academia and the software industry.

This study addresses the imperative for businesses in emerging markets to operate using sustainable practices and procedures. It establishes a clear connection between Change Control Board performance and sustainability, underlining that enhanced change management processes can lead to more sustainable practices. Our research is guided by the hypothesis that improved Change Control Board performance results in reduced resource consumption, waste generation, and environmental impact, which aligns with sustainable development objectives. By employing robust methodology and empirical data, the study demonstrates how more efficient change control processes contribute to enhanced sustainability, making businesses in emerging markets more environmentally responsible and economically efficient. Moreover, the findings support the significance of integrating sustainable practices into change management strategies, showcasing the potential for long-term cost savings and reputation enhancement. This work not only advances the understanding of Time to Market strategies but also provides actionable insights for businesses and policymakers seeking to promote sustainability in emerging markets.

5.1. Implications of the Study

This study addresses important gaps in the literature. It strengthens the project management literature by emphasizing the role of requirements uncertainty and its impact on the Time to Market of projects. There is great interest regarding agile methodology, but less consideration is given to issues of uncontrolled and untimely requirements and how they impact the Time to Market of software projects. Therefore, studying this particular area addresses the literature gap and provides substantial support to understanding the phenomenon, which is a reality on the ground. This study shows that requirements uncertainty has a significantly dominant impact on project performance and, subsequently, the Time to Market. On the other hand, the presence of the CCB was assumed to moderate and lessen the impact of requirements uncertainty, eventually reducing or perhaps controlling the Time to Market, and our study provides strong empirical evidence for this assumption. Organizations can focus on technology-driven tools, especially artificial intelligence-based tools, to improve CCB performance.

Similarly, this study has various practical implications in the field of project management. This research identified the importance of precise and accurate requirement gathering and the development of a work breakdown structure according to the requirements provided by the customer. The agile lifecycle approach is predominantly used in the IT industry, and we do acknowledge its benefits. However, this should not be reason to neglect the process of requirements gathering and defining the scope of projects. The results of this study suggest that project teams should engage with the customers as much as possible to determine the requirements and elaborate work packages to the maximum extent before the development work starts. We conclude that having an elaborate set of requirements at an earlier stage of the project reduces the uncertainty and eventually aids in improved performance of projects, subsequently reducing the Time to Market. This study will assist organizations to obtain a clear understanding of the impact of requirements uncertainty, and consequently, train people to work in the right direction while following an agile approach. In the IT industry, if the managers and team members of a project are sufficiently capable of engaging customers and can effectively manage the requirements-gathering process, they can deliver the desired results to the customers without delay in Time to Market. This will help in retaining the already existing customers as well as attracting new customers, thus developing a prominent position in the market. This study will serve both already-established firms and new startups because customers are the key entity of every business.

5.2. Conclusion and Recommendations

This study concludes that requirements uncertainty and Change Control Board performance have a significant impact on the Time to Market of software projects. Therefore, project managers and leaders should pay significant attention to the requirement-gathering and requirement-engineering phases to avoid any kind of uncertainty in project performance and to deliver the product well in time to the market. As shown by the findings of this study, the moderating role of CCBP in a prevailing RU situation is significant. This is logical as suggested by the practices and the contextual situational analysis. The CCB should perform adequately and vigilantly in the presence of higher uncertainty. Therefore, future researchers need to explore all the organizational or contextual (market) factors of product uncertainty. Similarly, future research is invited to improve this model by including other mediators and moderators like the role of product owners, quality assurance, customer engagements, etc.

The findings of this study have several implications. Firstly, they contribute to the project management literature by highlighting the significant impact of requirements uncertainty on TTM in software projects. While agile methodologies have received considerable attention, this study emphasizes the importance of effectively managing and controlling requirements to reduce TTM. Secondly, the study underscores the role of CCB performance as a moderator, suggesting that organizations should focus on improving CCB performance to mitigate the negative effects of requirement uncertainty on TTM. Employing technology-driven tools, particularly those based on artificial intelligence, can help enhance CCB performance.

Practically, this study emphasizes the importance of accurate requirement gathering and the development of a comprehensive work breakdown structure. While agile approaches are prevalent in the IT industry, our study highlights the need to prioritize thorough requirements gathering and scope definition. Engaging with customers extensively and clarifying requirements before project development commences can significantly reduce uncertainty and improve project performance, ultimately reducing TTM. This insight can guide organizations in training their project teams and adopting effective strategies to manage requirements within an agile framework.

In conclusion, this study fills a literature gap by providing empirical evidence on the impact of requirements uncertainty on TTM in software projects and the moderating role of CCB performance. The findings have theoretical implications for project management and practical implications for organizations in the IT industry. By addressing requirement uncertainty and improving CCB performance, organizations can enhance their project outcomes, meet customer expectations, and gain a competitive edge in the market.

Author Contributions: Conceptualization, I.H.-u.-R.; Validation, I.H.-u.-R.; Data curation, J.A.; Writing—original draft, S.I.; Writing—review & editing, M.G.K.; Supervision, J.A.; Funding acquisition, J.R.-S. and A.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Siegmund, N.; Ruckel, N.; Siegmund, J. Dimensions of Software Configuration: On the Configuration Context in Modern Software Development. In Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Virtual Event, 8–13 November 2020; ACM: New York, NY, USA, 2020.
2. Abbasi, A.Z. Specifying, estimating and validating consumer eSports engagement composite model: A composite confirmatory approach. *EuroMed J. Bus.* **2022**, *18*, 452–466. [[CrossRef](#)]
3. Khwaja, M.G.; Zaman, U.; Butt, A.H. Are digital influencers social change catalysts Empirical findings from the online apparel industry. *Int. J. Technol. Mark.* **2022**, *16*, 145. [[CrossRef](#)]
4. Espinel, G.P.; Medina, J.L.C.; Calero, M.J.F.; Urbieto, M. Software Configuration Management in Software Product Lines: Results of a Systematic Mapping Study. *IEEE Lat. Am. Trans.* **2022**, *20*, 718–730. [[CrossRef](#)]

5. Cooper, R.G.; Sommer, A.F. New-product portfolio management with agile: Challenges and solutions for manufacturers using agile development methods. *Res. Technol. Manag.* **2020**, *63*, 29–38. [CrossRef]
6. Kölln, A.-K.; Ongena, Y.P.; Aarts, K. The Effects of Sampling Frame Designs on Nonresponse and Coverage Error: Evidence from the Netherlands. *J. Surv. Stat. Methodol.* **2018**, *7*, 422–439. [CrossRef]
7. Henseler, J.; Ringle, C.M.; Sarstedt, M. A new criterion for assessing discriminant validity in variance-based structural equation modeling. *J. Acad. Mark. Sci.* **2015**, *43*, 115–135. [CrossRef]
8. Trunov, A.S.; Voronova, L.L.; Voronov, V.I.; Ayrapetov, D.P. Container Cluster Model Development for Legacy Applications Integration in Scientific Software System. In Proceedings of the 2018 IEEE International Conference “Quality Management, Transport and Information Security, Information Technologies” (IT&QM&IS), St. Petersburg, Russia, 24–28 September 2018; pp. 815–819.
9. Busari, S.A. Modelling and Analysing Software Requirements and Architecture Decisions under Uncertainty. Ph.D. Thesis, UCL, University College London, London, UK, 2019.
10. Godoy, C.; Pérez, A.J. Integration of Sensor and Actuator Networks and the Scada System to Promote the Migration of the Legacy Flexible Manufacturing System towards the Industry 4.0 Concept. *J. Sens. Actuator Netw.* **2018**, *7*, 23. [CrossRef]
11. Wilson, K.; Bell, C.; Wilson, L.; Witteman, H. Agile research to complement agile development: A proposal for an mHealth research lifecycle. *NPJ Digit. Med.* **2018**, *1*, 46. [CrossRef] [PubMed]
12. Rahman, A.; Williams, L. Characterizing Defective Configuration Scripts Used for Continuous Deployment. In Proceedings of the 2018 IEEE 11th International Conference on Software Testing, Verification and Validation (ICST), Västerås, Sweden, 9–13 April 2018; pp. 34–45.
13. Ke, L.; Gupta, U.; Cho, B.Y.; Brooks, D.; Chandra, V.; Diril, U.; Firoozshahian, A.; Hazelwood, K.; Jia, B.; Lee, H.-H.S.; et al. RecNMP: Accelerating Personalized Recommendation with near-Memory Processing. In Proceedings of the 2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA), Virtual Event, 30 May–3 June 2020.
14. Kirpitsas, I.K.; Pachidis, T.P. Evolution towards Hybrid Software Development Methods and Information Systems Audit Challenges. *Software* **2022**, *1*, 316–363. [CrossRef]
15. Hemalatha, C.; Sankaranarayanan, K.; Durairaj, N. Lean and agile manufacturing for work-in-process (WIP) control. *Mater. Today Proc.* **2021**, *46*, 10334–10338. [CrossRef]
16. Rakshit, S.; Islam, N.; Mondal, S.; Paul, T. Mobile apps for SME business sustainability during COVID-19 and onwards. *J. Bus. Res.* **2021**, *135*, 28–39. [CrossRef]
17. Diaz, A.; Schöggel, J.-P.; Reyes, T.; Baumgartner, R.J. Sustainable product development in a circular economy: Implications for products, actors, decision-making support and lifecycle information management. *Sustain. Prod. Consum.* **2021**, *26*, 1031–1045. [CrossRef]
18. Robillard, P.N. The role of knowledge in software development. *Commun. ACM* **1999**, *42*, 87–92. [CrossRef]
19. Leicht, D.; Castro-Fresno, D.; Díaz, J.; Baier, C. Multidimensional Construction Planning and Agile Organized Project Execution—The 5D-PROMPT Method. *Sustainability* **2020**, *12*, 6340. [CrossRef]
20. Moyano, C.G.; Pufahl, L.; Weber, I.; Mendling, J. Uses of business process modeling in agile software development projects. *Inf. Softw. Technol.* **2022**, *152*, 107028. [CrossRef]
21. Mahajan, H.V.P. *Modern Methods of Software Development Methodologies and Characteristics*; Svitla: Corte Madera, CA, USA, 2021.
22. Hoare, T.; Misra, J.; Leavens, G.T.; Shankar, N. The Verified Software Initiative: A Manifesto. In *Theories of Programming*; ACM: New York, NY, USA, 2021; pp. 81–92.
23. Abrahamsson, P.; Salo, O.; Ronkainen, J.; Warsta, J. Agile Software Development Methods: Review and Analysis. *arXiv* **2017**, arXiv:1709.08439.
24. Kleshchuk, S. Time to Market (TTM)—What Is It and Why Does It Matter for My Business? 2021. Available online: <https://enkonix.com/blog/time-to-market/> (accessed on 22 July 2023).
25. Uludağ, Ö.; Harders, N.-M.; Matthes, F. Documenting Recurring Concerns and Patterns in Large-Scale Agile Development. In Proceedings of the 24th European Conference on Pattern Languages of Programs, Irsee, Germany, 3–7 July 2019; ACM: New York, NY, USA, 2019.
26. Herrera, M.; Pérez-Hernández, M.; Parlikad, A.K.; Izquierdo, J. Multi-Agent Systems and Complex Networks: Review and Applications in Systems Engineering. *Processes* **2020**, *8*, 312. [CrossRef]
27. Fortino, G.; Savaglio, C.; Spezzano, G.; Zhou, M.C. Internet of Things as System of Systems: A Review of Methodologies, Frameworks, Platforms, and Tools. *IEEE Trans. Syst. Man Cybern. Syst.* **2021**, *51*, 223–236. [CrossRef]
28. Malik, M.F.; Khan, M.A. “Tracking Engagement through Leader” Authentic Leadership’s Consequences on Followers’ Attitudes: A Sequential Mediated Mode. *Int. J. Public Adm.* **2019**, *43*, 831–838. [CrossRef]
29. Mohan, K.; Xu, P.; Cao, L.; Ramesh, B. Improving change management in software development: Integrating traceability and software configuration management. *Decis. Support Syst.* **2008**, *45*, 922–936. [CrossRef]
30. Sjøberg, D.I.K. An Empirical Study of WIP in Kanban Teams. In Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, Lund, Sweden, 19–20 September 2012; ACM: New York, NY, USA, 2018.
31. Haleem, M.; Farooqui, F. Faisal Tackling Requirements Uncertainty in Software Projects: A Cognitive Approach. *Int. J. Cogn. Comput. Eng.* **2021**, *2*, 180–190. [CrossRef]

32. Shih-Chiehhsu, J.C. The Impacts of User Review on Software Responsiveness: Moderating Requirements Uncertainty. *Inf. Manag.* **2008**, *45*, 203–210.
33. Jayatilleke, S.; Lai, R. A systematic review of requirements change management. *Inf. Softw. Technol.* **2018**, *93*, 163–185. [[CrossRef](#)]
34. Singh, O.; Panwar, S.; Kapur, P.K. Determining Software Time-to-Market and Testing Stop Time when Release Time is a Change-Point. *Int. J. Math. Eng. Manag. Sci.* **2020**, *5*, 208–224. [[CrossRef](#)]
35. Chernyak-Hai, L.; Rabenu, E. The New Era Workplace Relationships: Is Social Exchange Theory Still Relevant? *Ind. Organ. Psychol.* **2018**, *11*, 456–481. [[CrossRef](#)]
36. Scot, M. *Clinical Informatics Board Review and Self Assessment*; Springer: Berlin, Germany, 2018.
37. Fahmy, S.; Deraman, A.; Yahaya, J.H. The Role of Human in Software Configuration Management. In Proceedings of the 2018 7th International Conference on Software and Computer Applications, Kuantan, Malaysia, 8–10 February 2018; ACM: New York, NY, USA, 2018.
38. Abbasi, A.Z.; Azeem, S.; Farooq, M.U.; Hussain, K.; Ting, D.H.; Rehman, U.; Griffiths, M.D.; Pakpour, A.H. Engagement in educational games and quality of life in early and middle childhood: Evidence from a developing country. *Curr. Psychol.* **2022**, *42*, 19386–19400. [[CrossRef](#)]
39. Zaman, U.; Nawaz, S.; Tariq, S.; Humayoun, A.A. Linking Transformational Leadership and “Multi-Dimensions” of Project Success: Moderating Effects of Project Flexibility and Project Visibility Using PLS-SEM. *Int. J. Manag. Projects Bus.* **2019**, *13*, 103–127. [[CrossRef](#)]
40. Hameed, A.; Khwaja, M.G.; Zaman, U. Configuring optimal contextual performance and task performance in offshore business processing organizations. *Bus. Process. Manag. J.* **2023**, *29*, 285–307. [[CrossRef](#)]
41. Turi, J.A.; Khwaja, M.G.; Tariq, F.; Hameed, A. The role of big data analytics and organizational agility in improving organizational performance of business processing organizations. *Bus. Process. Manag. J.* **2023**, *29*, 2081–2106. [[CrossRef](#)]
42. Malik, M.F. “Perfectionism is a debacle” how a perfectionist leader hinders in business processes? A multiple mediated model. *Bus. Process. Manag. J.* **2023**, *29*, 1184–1203. [[CrossRef](#)]
43. Aktan, M.; Anjam, M.; Zaman, U.; Khwaja, M.G.; Akram, U. Missing link in ‘new-normal’ for higher education: Nexus between online experiential marketing, perceived-harm, social distancing concern and university brand evangelism in China. *J. Mark. High. Educ.* **2023**, 1–26. [[CrossRef](#)]
44. Zaman, U. Seizing Momentum on Climate Action: Nexus between Net-Zero Commitment Concern, Destination Competitiveness, Influencer Marketing, and Regenerative Tourism Intention. *Sustainability* **2023**, *15*, 5213. [[CrossRef](#)]
45. Zaman, U. Editorial: Metaverse going beyond adoption: The next frontier for global healthcare. *Front. Public Health* **2023**, *11*, 1194285. [[CrossRef](#)]
46. Malik, M.F.; Khan, M.A.; Mahmood, S. Increasing the efficiency of business process through authentic leaders and follower’s attitude. *Bus. Process. Manag. J.* **2021**, *27*, 529–545. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.