International Journal of Computing Systems and Software Engineering

https://ijcsse.net

Requirements Volatility for Project Delays in Outsourced Software Contracts

Elijah Reed[·]

Department of Advanced Computing, Viterbi School of Engineering, University of Southern California, Los Angeles, California, USA

Corresponding author: elijah.reed@gmail.com

Abstract

The pervasive phenomenon of requirements volatility remains a primary catalyst for schedule overruns in outsourced software development projects. While cross-sectional studies have extensively documented the general correlation between changing requirements and project delays, there is a critical shortage of longitudinal research capable of capturing the dynamic, time-dependent nature of these disruptions. This paper addresses this substantial research gap by investigating how requirements volatility, mapped continuously across the software development lifecycle, influences project delays within the context of outsourced contracts. Utilizing a comprehensive longitudinal tracking methodology, this study observes empirical data drawn from enterprise-level software projects over an extended timeframe. By operationalizing requirements volatility into distinct sub-categories including additions, modifications, and deletions, the research identifies critical phases where changes exert the most severe impact on delivery schedules. Furthermore, the analysis explores the moderating effects of different contract structures, specifically fixed-price versus time-and-materials agreements, on the relationship between volatility and schedule slippage. The findings reveal that late-stage modifications in fixed-price contracts generate disproportionately high negotiation overhead, thereby exacerbating delays beyond pure technical rework. The insights provided offer significant theoretical contributions to software project management literature and deliver actionable intelligence for practitioners seeking to mitigate schedule risks through adaptive contractual frameworks and continuous monitoring mechanisms.

Keywords: Requirements Volatility; Software Outsourcing; Longitudinal Tracking; Project Delays; Contract Management

1. Introduction

1.1 The Landscape of Outsourced Software Development

The global landscape of software engineering has undergone a profound transformation over the past three decades, driven largely by the strategic imperative to leverage distributed talent pools and optimize operational expenditures. Outsourced software development has consequently evolved from a peripheral cost-saving tactic into a core strategic mechanism for organizations worldwide. Through outsourcing, enterprises can access highly specialized technical competencies that may be scarce or cost-prohibitive within their domestic labor markets. However, the dispersion of development activities across organizational and geographical boundaries introduces unprecedented levels of complexity in project governance [1]. The geographical and cultural distances inherent in these arrangements frequently exacerbate communication friction, leading to profound misalignments between client expectations and vendor deliverables. In such distributed environments, the formal contract becomes the primary instrument for aligning incentives, defining scope, and allocating risk. Yet, the inherent uncertainty of software engineering severely limits the efficacy of rigid, upfront contractual agreements. As projects progress and business environments evolve, initial specifications inevitably become obsolete or require substantial refinement. This fundamental tension between the static nature of traditional legal contracts and the dynamic reality of software development creates a fertile ground for project management failures, most notably manifested as severe schedule overruns [2]. The resulting delays not only incur direct financial penalties but also inflict strategic damage

through missed market opportunities and compromised competitive advantages. Understanding the precise mechanisms through which these delays materialize within outsourced contexts therefore represents a critical imperative for both academic researchers and industry practitioners [3].

1.2 The Phenomenon of Requirements Volatility

At the core of software project instability lies the phenomenon of requirements volatility, broadly defined as the tendency of system requirements to change over the duration of the development lifecycle. Unlike physical construction projects where architectural blueprints can be definitively finalized prior to execution, software systems are fundamentally malleable and inextricably linked to fluid business processes [4]. Requirements volatility is not necessarily a symptom of poor initial planning; rather, it often reflects a rational organizational response to shifting market conditions, emerging technological capabilities, or evolving user feedback. Changes can manifest in various forms, including the introduction of entirely new functional requirements, the substantial modification of existing specifications, or the deletion of obsolete features [5]. While embracing change is a core tenet of modern agile methodologies, uncontrolled volatility in an outsourced context introduces severe cascading disruptions. When a requirement changes, the impact propagates through the entire engineering pipeline. Developers must discard completed code, architects must reassess system integration points, and quality assurance teams must redesign testing protocols. In an outsourced environment, this technical rework is compounded by administrative overhead [6]. Every substantial change necessitates a formal impact analysis, renegotiation of effort estimates, and formal contract amendments. This dual burden of technical rework and administrative friction creates a compounding delay effect that frequently derails project schedules. The later in the lifecycle a change occurs, the more exponentially expensive and time-consuming it is to implement, as the foundation upon which the software is built has already hardened into complex code structures [7].

1.3 Research Gap and Objectives

Despite the universally acknowledged criticality of requirements volatility, the existing corpus of software engineering literature exhibits a pronounced methodological limitation. The overwhelming majority of prior investigations rely on cross-sectional survey data or post-mortem project analyses. These retrospective approaches treat requirements volatility as a static, aggregate variable, typically measured as a single percentage of requirements altered by the end of the project [8]. Such methodologies fundamentally fail to capture the temporal dynamics of change. A project experiencing consistent, minor refinements throughout its early phases presents a vastly different risk profile compared to a project encountering massive, structural changes during the final stages of user acceptance testing. Yet, traditional static metrics would render these two distinct scenarios statistically identical [9]. To truly comprehend the mechanics of project delay, researchers must track volatility longitudinally, observing exactly when changes occur, how long they take to process, and how they accumulate over time [10]. This study aims to bridge this substantial gap by providing a comprehensive longitudinal analysis of requirements volatility within outsourced software contracts. The primary objectives are threefold. First, to establish an empirical baseline of how different types of requirements changes distribute across various phases of the software development lifecycle. Second, to quantify the specific delay impact of these changes based on their temporal occurrence. Third, to analyze how different outsourcing contract structures moderate the relationship between requirements volatility and project delays. By addressing these objectives, this research endeavors to construct a more granular, time-aware model of software project disruption [11].

2. Literature Review and Theoretical Framework

2.1 Dynamics of Software Outsourcing Contracts

The governance of outsourced software development is fundamentally rooted in contract theory, specifically transaction cost economics and agency theory. Transaction cost economics posits that the optimal governance structure for an exchange depends on the level of asset specificity, uncertainty, and transaction frequency [12]. Software development is characterized by extremely high uncertainty, making comprehensive contracting virtually impossible. Agency theory further highlights the inherent conflict of interest between the principal, the client organization, and the agent, the software vendor. The client typically seeks maximum functionality for a minimum price, while the vendor aims to maximize profit margins by minimizing effort [13]. To navigate these theoretical tensions, the industry primarily relies on two archetypal contract structures: fixed-price and time-and-materials. Fixed-price contracts provide the client with budget

certainty by shifting the financial risk of cost overruns entirely onto the vendor. Consequently, vendors utilize these contracts only when requirements are perceived to be stable and clearly defined. In contrast, time-and-materials contracts shift the risk back to the client, as the vendor is compensated for all hours worked regardless of the final output [14]. This structure inherently accommodates changing requirements but lacks strict financial boundaries. The literature suggests that the choice of contract profoundly influences how requirements volatility is managed [15]. Under fixed-price arrangements, vendors heavily resist changes to protect their margins, leading to protracted negotiations and change control procedures that introduce significant administrative delays. Conversely, while time-and-materials contracts allow for more fluid technical implementation of changes, they can suffer from scope creep and a lack of urgency, equally contributing to schedule slippage [16].

2.2 Conceptualizing Requirements Volatility

Requirements engineering literature provides a robust framework for conceptualizing the diverse nature of requirements volatility. Scholars have long recognized that change is not a monolithic construct but rather a multifaceted phenomenon driven by disparate underlying causes [17]. Volatility can be broadly categorized into endogenous and exogenous changes. Endogenous changes originate from within the project ecosystem, often resulting from initial misunderstandings, poor requirements elicitation, or technical unfeasibility discovered during the implementation phase. Exogenous changes, on the other hand, are driven by external factors beyond the immediate control of the project team, such as new regulatory compliance mandates, shifts in competitor strategies, or macroeconomic fluctuations altering the client business model [18]. Furthermore, volatility can be classified by its operational manifestation. Additions involve the introduction of entirely new functional scope. Deletions involve the removal of planned features, which, while seemingly reducing effort, can cause integration failures if underlying architectural dependencies have already been established. Modifications involve altering the logic, performance criteria, or user interface of an existing requirement [19]. Each operational type demands a unique cognitive and technical response from the engineering team. Previous theoretical models suggest that modifications are often the most disruptive, as they require developers to unlearn previous assumptions and carefully untangle existing code without breaking adjacent functionalities [20]. Understanding these typologies is essential for developing precise tracking mechanisms that can correlate specific types of volatility with corresponding magnitudes of project delay.

2.3 Project Delay Mechanisms and Ripple Effects

The translation of a changed requirement into a project delay is a complex mechanical process characterized by non-linear ripple effects. When a change request is formally approved, the immediate technical effort required to write the new code represents only a fraction of the total delay incurred [21]. The initial disruption occurs during the analysis phase, where senior architects and analysts must pause ongoing development to evaluate the impact of the proposed change on the broader system architecture. This context-switching imposes a severe cognitive penalty on the development team, breaking momentum and reducing overall productivity. Once development begins, the phenomenon of architectural degradation often emerges [22]. Software systems are typically designed to optimally solve the initial set of requirements. Continuous modifications force developers to compromise the original architectural integrity, introducing technical debt and increasing the fragility of the codebase. This fragility exponentially increases the time required to perform subsequent tasks, creating a compounding delay mechanism. Furthermore, the ripple effect extends heavily into the quality assurance domain. A single modified requirement can invalidate hundreds of automated test scripts, necessitating extensive rework of the testing framework itself [23]. In outsourced environments, these technical ripple effects are amplified by communication latency. Queries regarding ambiguous changes must be routed back to the client, often across different time zones, resulting in days of idle wait time for the development staff [24]. This comprehensive understanding of delay mechanisms emphasizes why static measurements of volatility are insufficient.

2.4 The Imperative for Longitudinal Tracking

The prevailing reliance on cross-sectional methodologies in requirements volatility research has significantly constrained theoretical advancement in the field. Cross-sectional studies essentially capture a single snapshot of a project upon its conclusion, aggregating all volatility events into a cumulative metric [25]. This methodological paradigm obscures the critical dimension of time. Software engineering is a phased,

evolutionary process. The impact of a requirement change introduced during the initial design phase is fundamentally distinct from the impact of an identical change introduced during the final stages of system integration. Early changes often require only the revision of documentation and conceptual models, incurring minimal delay. Late changes require the dismantling of compiled code, profound regression testing, and extensive redeployment procedures, incurring maximum delay [26]. Longitudinal tracking provides the vital mechanism to capture this temporal heterogeneity. By continuously monitoring the project environment at regular intervals, researchers can construct detailed time-series data that maps exactly when volatility events occur and tracks their direct downstream consequences on schedule milestones. This continuous observation enables the application of advanced statistical modeling to isolate the precise delay impact of changes across different project lifecycle phases. Furthermore, longitudinal data permits the observation of team adaptation over time, revealing whether outsourced vendors develop increased resilience to volatility as they gain familiarity with the client business domain [27]. The transition toward longitudinal tracking represents a necessary methodological evolution to produce insights that accurately reflect the complex reality of modern software development.

3. Methodology and Longitudinal Research Design

3.1 Data Collection and Sampling Strategy

To capture the dynamic interplay between requirements volatility and project delays, this study employed a rigorous longitudinal research design focused on empirical data extracted from real-world outsourced software projects. The target population comprised enterprise-level software development contracts executed between large corporate clients and specialized offshore software vendors. To ensure data comparability and consistency, the sampling strategy was restricted to projects that utilized formal project management and issue tracking software, enabling the precise, time-stamped extraction of requirement modifications. The final dataset consisted of seventy-five distinct software projects tracked continuously over a period of thirty-six months. These projects spanned diverse industry sectors, including financial services, healthcare informatics, and retail supply chain management, thereby providing a broad perspective on business-driven volatility. To control for size effects, all selected projects were classified as large-scale, characterized by initial budgets exceeding one million dollars and scheduled durations of no less than twelve months. The inclusion criteria strictly required that both the initial contractual agreements and all subsequent change request documentation were accessible for academic review. This level of access permitted researchers to triangulate quantitative metrics extracted from digital tracking systems with qualitative context derived from formal contractual amendments. The continuous observation window allowed the research team to monitor the lifecycle of each requirement from initial elicitation through development, testing, and final deployment.

3.2 Operationalization of Variables

The precision of this longitudinal study relied heavily on the rigorous operationalization of both independent and dependent variables. The primary dependent variable was Project Schedule Delay. Rather than measuring delay merely as the final variance between the planned and actual completion dates, delay was operationalized longitudinally. It was recorded as the cumulative deviation in days from the baseline schedule at multiple predefined project milestones, specifically the completion of architectural design, the conclusion of core development, the end of user acceptance testing, and final deployment. This multi-point measurement allowed for the calculation of delay velocity across different project phases. The independent variable, Requirements Volatility, was deconstructed into precise, quantifiable metrics. Volatility was tracked not as a static percentage, but as a continuous time-series count of volatility events. A volatility event was defined as any formally logged addition, deletion, or modification of a system requirement. Furthermore, to capture the temporal dimension, the project lifecycle was divided into four standardized quartiles based on elapsed scheduled time. Every volatility event was time-stamped and assigned to the quartile in which it occurred. In addition to temporal occurrence, events were categorized by their operational type. Additions were measured by the estimated function points of the newly introduced scope. Deletions were measured by the function points of the removed scope. Modifications were assessed based on a complexity scale derived from the number of interconnected system modules affected. To isolate the impact of volatility, several control variables were meticulously tracked. These included the total initial size of the project

measured in total function points, the aggregate experience level of the vendor development team measured in years of domain expertise, and the specific structure of the outsourcing contract utilized.

Table 1: Descriptive Statistics of Tracked Projects

Project Category	Number of Projects	Average Duration (Months)	Average Volatility Events	Average Total Delay (Days)
Financial Systems	25	18.5	142	45.2
Healthcare Informatics	20	22.0	185	62.8
Retail Supply Chain	30	14.2	110	28.4

3.3 Tracking Mechanisms and Analytical Approach

The extraction and processing of the longitudinal data required a highly systematic tracking mechanism, leveraging the digital exhaust generated by modern software engineering tools. The research team utilized specialized data extraction scripts to pull historical logs directly from the project management repositories of the participating vendor organizations. These repositories contained the entire historical record of all tasks, bug reports, and requirement specifications. By analyzing the revision history of the requirement artifacts, the system could automatically detect the exact timestamp when a specification was altered, thereby registering a volatility event. The textual description of the change was then subjected to natural language processing algorithms to automatically categorize the event as an addition, deletion, or modification, which was subsequently verified by human researchers to ensure high classification accuracy. To correlate these volatility events with project delays, the tracking mechanism cross-referenced the requirement alteration timestamps with the completion dates of associated project tasks. If a task linked to a modified requirement missed its original baseline completion date, the magnitude of the slip was recorded and attributed to that specific volatility event.

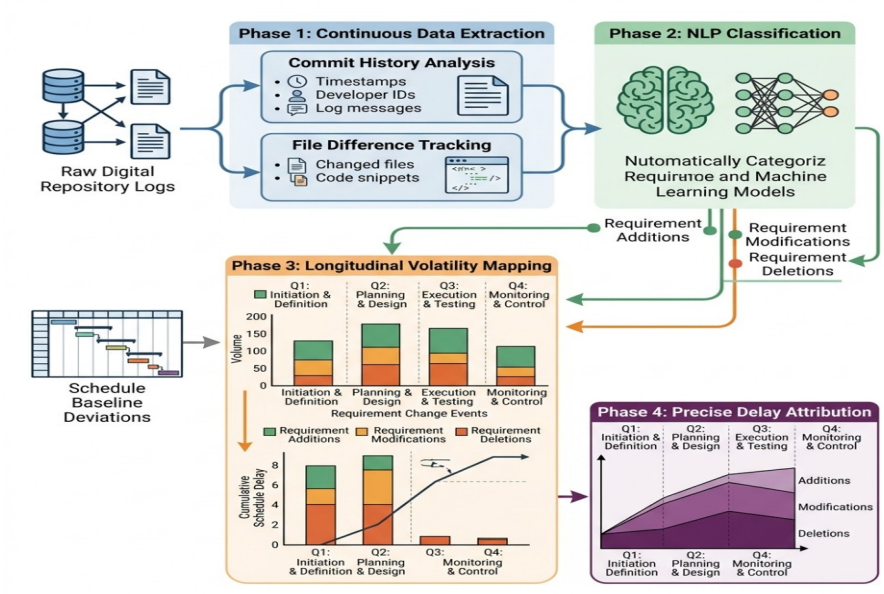

Figure 1: Comprehensive Schematic of Longitudinal Data Extraction and Volatility Mapping Mechanism

The analytical approach utilized advanced panel data regression techniques, specifically fixed-effects models, to analyze the time-series data. This statistical method was selected because it controls for unobserved heterogeneity across the different software projects, ensuring that the estimated impact of volatility is not confounded by unique project-specific characteristics that remain constant over time. The models were designed to estimate the marginal delay impact of a single volatility event, conditional upon the project phase in which it occurred and the type of change it represented. Furthermore, interaction terms were introduced into the regression equations to explicitly test the moderating effect of contract types. By comparing the coefficients of the interaction between volatility and fixed-price contracts against the interaction between volatility and time-and-materials contracts, the analytical framework could robustly quantify how legal governance structures exacerbate or mitigate the mechanical delays caused by shifting requirements.

4. Empirical Results and Impact Analysis

4.1 Temporal Distribution of Volatility Trends

The longitudinal tracking mechanism yielded profound insights into the temporal distribution of requirements volatility across the software development lifecycle. Contrary to the idealized waterfall model, which assumes that requirements can be entirely frozen before development commences, the empirical data demonstrated that volatility is a continuous, ubiquitous presence. However, the intensity and nature of these changes exhibited distinct, statistically significant patterns when mapped across the four temporal quartiles of the projects. During the first quartile, which predominantly encompasses design and initial prototyping, volatility events were characterized by exceptionally high frequency but relatively low complexity. The majority of these early events were minor modifications and clarifications to existing requirements, reflecting the natural process of mutual learning as the client and vendor aligned their mental models of the proposed system. As projects transitioned into the second and third quartiles, representing the core coding and integration phases, the absolute frequency of volatility events exhibited a measurable decline. However, the operational type of volatility shifted dramatically. This middle phase was dominated by requirement additions, often driven by external business stakeholders realizing that the initial scope lacked critical functionalities needed to achieve the underlying business objectives. The fourth quartile, representing late-stage system integration and user acceptance testing, revealed a deeply concerning trend. While the total number of changes was lowest in this final phase, the events that did occur were almost exclusively massive structural modifications and complex feature deletions. These late-stage changes were primarily reactionary, resulting from the client organization interacting with the near-completed software and discovering profound misalignments between the constructed system and actual operational workflows. This temporal mapping clearly established that volatility does not decrease in a linear fashion; rather, it evolves in operational complexity as the project matures.

4.2 Schedule Slippage and Phase-Dependent Impacts

The application of fixed-effects regression models to the longitudinal dataset provided robust empirical confirmation of the phase-dependent nature of project delays. The statistical analysis revealed a strong, positive, and non-linear correlation between the timing of requirements volatility and the magnitude of subsequent schedule slippage. When volatility events occurred during the first quartile of the project lifecycle, their impact on the final delivery schedule was found to be marginal and, in many statistical iterations, virtually negligible. The development teams demonstrated a high capacity to absorb early changes through minor adjustments to architectural blueprints and database schemas without disrupting critical path activities. However, as volatility events were introduced later in the lifecycle, their delay impact escalated exponentially. Changes introduced during the third quartile, the peak coding phase, generated moderate to severe schedule deviations. The analysis indicated that during this phase, a single major requirement modification necessitated extensive code refactoring, which subsequently triggered cascading delays as dependent modules had to be paused and redesigned. The most catastrophic delays were inextricably linked to fourth-quartile volatility. Modifications requested during final user testing produced delay multipliers that were exponentially higher than identical changes requested during the design phase. The data clearly demonstrated that late-stage changes entirely invalidated prior quality assurance efforts. The development teams were forced not only to write new code but also to comprehensively reverse-engineer existing integrations, rewrite hundreds of automated testing scripts, and repeat laborious security validation protocols. The regression output definitively proved that measuring the pure volume of requirements change is fundamentally insufficient for predicting schedule overruns; the precise temporal location of the change is the dominant variable determining the severity of the delay.

4.3 The Moderating Influence of Contractual Structures

Beyond the mechanical and technical causes of delay, the longitudinal data facilitated a deep examination of how legal governance structures moderate the impact of requirements volatility. The statistical interaction models comparing fixed-price contracts to time-and-materials contracts revealed profound operational differences in how change is processed. In projects governed by strict fixed-price agreements, the introduction of requirements volatility consistently resulted in significantly longer overall project delays compared to similar volatility in time-and-materials projects. This discrepancy was not attributable to technical incompetence, but rather to severe administrative and negotiation friction. Under fixed-price

conditions, every proposed change request triggered a defensive organizational response from the vendor, who viewed the change as a direct threat to profit margins. The empirical tracking showed that in these environments, development work on the affected modules would completely halt while legal and management teams engaged in protracted negotiations regarding the cost and schedule impact of the proposed change. This administrative latency often consumed more calendar days than the actual technical implementation of the code. Conversely, time-and-materials contracts exhibited far less administrative friction. Because vendors were compensated for all hours expended, they readily accepted and implemented changes without requiring extensive contractual renegotiation. However, the data also indicated a secondary risk associated with time-and-materials arrangements. While they avoided negotiation delays, they frequently suffered from uncontrolled scope creep, leading to a steady accumulation of minor additions that gradually inflated the total project duration. Ultimately, the empirical results suggest that fixed-price contracts artificially amplify the schedule impact of volatility by overlaying severe transaction costs onto the necessary technical rework.

5. Discussion and Conclusion

5.1 Theoretical and Practical Implications

The findings derived from this comprehensive longitudinal tracking study deliver substantial contributions to the theoretical foundations of software engineering management and contract theory. By definitively demonstrating the non-linear, phase-dependent impact of requirements volatility, this research challenges the utility of static, cross-sectional metrics that have long dominated the literature. The evidence necessitates a paradigm shift in how academic models conceptualize software project risk, moving away from aggregate volume measurements toward time-series analysis that respects the evolutionary nature of software construction. Furthermore, by illuminating the profound moderating effect of contract structures, this study bridges the gap between project management theory and transaction cost economics, proving that administrative friction is a measurable and dominant component of project delay. From a practical perspective, these insights offer critical, actionable intelligence for client organizations and software vendors. Project management offices must abandon the illusion that rigid initial planning can eliminate requirements volatility. Instead, they must implement continuous, automated tracking mechanisms that flag the temporal occurrence of changes in real-time. For practitioners, the data strongly advocates against the use of strict fixed-price contracts for large, complex software initiatives where business requirements are fluid. The administrative overhead of managing change under such rigid structures virtually guarantees severe schedule overruns. Organizations are strongly advised to adopt hybrid contracting models that combine the budget boundaries of fixed-price arrangements with predefined, agile mechanisms for managing scope modifications without requiring exhaustive legal renegotiation for every functional adjustment.

5.2 Limitations and Avenues for Future Research

While the longitudinal methodology provides a significant advancement over previous observational techniques, this study is not without inherent limitations that must be acknowledged. The primary limitation resides in the sample composition. Although the dataset of seventy-five enterprise projects is robust for longitudinal analysis, it was intentionally restricted to large-scale initiatives executed by specialized offshore vendors. The observed dynamics of volatility and delay may operate differently within the context of small-scale, domestic outsourcing arrangements, or within open-source collaborative environments that lack formal legal contracts. Additionally, while the automated extraction of data from project management repositories ensures high quantitative accuracy regarding the timing of events, it may fail to capture informal, undocumented changes that occur through direct, unrecorded communication between client stakeholders and lead developers. Future research endeavors should aim to expand the longitudinal tracking methodology to encompass a broader spectrum of project sizes and engagement models. Furthermore, there is a critical need for academic exploration into the efficacy of specific agile contracting frameworks, such as target-cost contracts or capacity-based agreements, to determine if they successfully mitigate the administrative friction identified in traditional fixed-price arrangements during periods of high volatility.

5.3 Final Conclusion

In conclusion, this research comprehensively establishes that requirements volatility is not a static risk factor, but a dynamic, time-dependent catalyst for project delays in outsourced software contracts. Through

rigorous longitudinal tracking, the study proves that the schedule impact of a changing requirement is fundamentally dictated by its operational complexity and the specific lifecycle phase in which it is introduced. Late-stage structural modifications introduce catastrophic delays not merely through the necessity of complex technical refactoring, but through the profound invalidation of established architectural dependencies and quality assurance frameworks. Crucially, the research demonstrates that these technical challenges are severely exacerbated by the governance structure of the outsourcing agreement. Rigid fixed-price contracts interact antagonistically with fluid requirements, generating massive administrative and negotiation delays that far exceed the temporal cost of actual software development. To successfully navigate the inherent uncertainties of modern software engineering, both client and vendor organizations must transition toward adaptive governance models and continuous monitoring systems. Only by acknowledging and structurally accommodating the inevitability of continuous change can the software industry begin to systematically reduce the pervasive prevalence of costly and strategic schedule overruns.

References

1. Avvaru, V.; Bruno, G.; Chiabert, P.; Traini, E. Integration of PLM, MES and ERP Systems to Optimize the Engineering, Production and Business. In Proceedings of the 17th International Conference on Product Lifecycle Management-PLM-Annual, Rapperswil, Switzerland, 5–8 July 2020; pp. 70–82.

2. Paul, S.G.; Saha, A.; Arefin, M.S.; Bhuiyan, T.; Biswas, A.A.; Reza, A.W.; Alotaibi, N.M.; Alyami, S.A.; Moni, M.A. A Comprehensive Review of Green Computing: Past, Present, and Future Research. IEEE Access 2023, 11, 87445–87494.

3. Peter, M. K., Kraft, C., & Lindeque, J. (2020). Strategic action fields of digital transformation: An exploration of the strategic action fields of Swiss SMEs and large enterprises. Journal of Strategy and Management, 13(1), 160–180.

4. Recker, J., Holten, R., Hummel, M., & Rosenkranz, C. (2017). How agile practices impact customer responsiveness and development success: A field study. Project Management Journal, 48(2), 99–121.

5. Chen, M., Martins, T. S., Zhang, L., & Dong, H. (2025). Digital transformation in project management: A systematic review and research agenda. Systems, 13(8), 625.

6. Mutoni, U. (2025). Influence of digital transformation on project management efficiency in construction projects in Rwanda. Journal of Entrepreneurship and Project Management, 10(2), 47–57.

7. Georgiou, S.; Rizou, S.; Spinellis, D. Software Development Lifecycle for Energy Efficiency: Techniques and Tools. ACM Comput. Surv. 2019, 52, 1–33.

8. Hidalgo, M.; Yanine, F.; Paredes, R.; Frez, J.; Solar, M. What Is the Process? A Metamodel of the Requirements Elicitation Process Derived from a Systematic Literature Review. Processes 2025, 13, 20.

9. Sepúlveda, S.; Cravero, A.; Fonseca, G.; Antonelli, L. Systematic Review on Requirements Engineering in Quantum Computing: Insights and Future Directions. Electronics 2024, 13, 2989.

10. Huber, S.; Lorey, T.; Felderer, M. Techniques for Improving the Energy Efficiency of Mobile Apps: A Taxonomy and Systematic Literature Review. In Proceedings of the 2023 49th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), Durres, Albania, 6–8 September 2023; pp. 286–292.

11. Zhao, L.; Alhoshan, W.; Ferrari, A.; Letsholo, K.J.; Ajagbe, M.A.; Chioasca, E.V.; Batista-Navarro, R.T. Natural Language Processing for Requirements Engineering: A Systematic Mapping Study. ACM Comput. Surv. 2021, 54, 4689.

12. FAO. How to Feed the World 2050. High Level Expert Forum—How to Feed the World in 2050. Office of the Director, Agricultural Development Economics Division, Rome 12–13 October 2009. Available online: https://www.fao.org/fileadmin/templates/wsfs/docs/expert_paper/How_to_Feed_the_World_in_2050.pdf (accessed on 15 January 2026).

13. Antonijević, M.; Bradić-Martinović, A.; Banović, J.; Ivanović, Đ. Is There a Relationship Between Country Development and Citizens' Level of Digital Skills? Econ. Anal. J. Emerg. Econ. 2023, 56, 57–68.

14. Corbalan, L.; Fernandez, J.; Cuitiño, A.; Delia, L.; Cáseres, G.; Thomas, P.; Pesado, P. Development frameworks for mobile devices: A comparative study about energy consumption. In Proceedings of the 5th International Conference on Mobile Software Engineering and Systems (MOBILESoft '18), Gothenburg, Sweden, 27–28 May 2018; pp. 191–201.

15. Moshnyaga, V.G. Lifecycle energy assessment of mobile applications. In Proceedings of the 2013 International Workshop on Software Development Lifecycle for Mobile (DeMobile 2013), Saint Petersburg, Russia, 19 August 2013; pp. 17 –23.

16. Nunkesser, R. Beyond web/native/hybrid: A new taxonomy for mobile app development. In Proceedings of the 5th International Conference on Mobile Software Engineering and Systems (MOBILESoft '18), Gothenburg, Sweden, 27–28 May 2018; pp. 214–218.

17. Bierwolf, R. (2016). Project excellence or failure? Doing is the best kind of learning. IEEE Engineering Management Review, 44(2), 26–32.

18. May, G.; Stahl, B.; Taisch, M.; Kiritsis, D. Energy management in manufacturing: From literature review to a conceptual framework. J. Clean. Prod. 2017, 167, 1464–1489.

19. Alshammari, B.; Singh, M.M. A Systematic Literature Review on Tackling Cyber Threats for Cyber Logistic Chain and Conceptual Frameworks for Robust Detection Mechanisms. IEEE Access 2025, 13, 67661–67692.

20. Lokuge, S., Sedera, D., Grover, V., & Dongming, X. (2019). Organizational readiness for digital innovation: Development and empirical calibration of a construct. Information & Management, 56(3), 445–461.

21. Warner, K. S. R., & Wäger, M. (2019). Building dynamic capabilities for digital transformation: An ongoing process of strategic renewal. Long Range Planning, 52(3), 326–349.

22. Buh, B., Kovačič, A., & Indihar Štemberger, M. (2015). Critical success factors for different stages of business process management adoption—A case study. Economic Research-Ekonomska Istraživanja, 28(1), 243–258.

23. Benlian, A., & Haffke, I. (2016). Does mutuality matter? Examining the bilateral nature and effects of CEO–CIO mutual understanding. Journal of Strategic Information Systems, 25(2), 104–126.

24. Brauner, P., & Ziefle, M. (2022). Beyond playful learning—Serious games for the human-centric digital transformation of production and a design process model. Technology in Society, 71, 102140.

25. Petersen, K.; Feldt, R.; Mujtaba, S.; Mattsson, M. Systematic Mapping Studies in Software Engineering. In Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering (EASE), Bari, Italy, 26–27 June 2008; pp. 1–10.

26. Osorio-Gómez, C. C., Herrera, R. F., Tupėnaitė, L., & Pellicer, E. (2025). The digital organization transformation (DOT) model: Bridging digital transformation and organizational structures in construction firms. Journal of Civil Engineering and Management, 31(8), 909–925.

27. Grant, M.J.; Booth, A. A typology of reviews: An analysis of 14 review types and associated methodologies. Health Inf. Libr. J. 2009, 26, 91–108.