*software*

MDPI

*Article*

# Development Bursts, Socio-Technical Congruence, and Delivery Reliability in Distributed Open-Source Software Projects

**Godfried B. Adaba**

Royal Docks School of Business and Law, University of East London, Docklands Campus, University Way, London E16 2RD, UK; g.adaba@uel.ac.uk

**Abstract**

Socio-technical congruence (STC) theory posits that software development effectiveness depends on the degree to which developer coordination aligns with technical requirements. However, much existing research treats congruence as a static project attribute, overlooking the dynamic coordination needs that arise during periods of intensive development. This study addresses this gap by examining development bursts, brief episodes of significant technical change and collaboration that alter coordination demands. Using fixed-effects regression models, the analysis examines 45,981 development bursts from 6401 open-source software projects. The findings show that higher burst intensity is positively associated with both congruence and delivery throughput, suggesting that effective collaboration often centers on shared integration challenges. Conversely, higher coordination complexity is associated with lower congruence and reduced delivery reliability, indicating that the main risks associated with bursts stem from structural rather than purely volumetric factors. Team experience also mitigates the negative effects of complexity on congruence, highlighting the importance of accumulated coordination capability. However, after accounting for demand-side factors, the relationship between congruence and delivery reliability becomes negative, indicating that congruence observed during bursts reflects not only alignment quality but also coordination burden. These findings extend STC theory by reconceptualizing congruence as a dynamic state and emphasizing the need to understand its evolution in high-frequency coordination contexts.

**Keywords:** socio-technical congruence; development bursts; distributed software development; coordination complexity; delivery reliability; open-source software; mining software repositories



Academic Editor: Paulo Ferreira

Received: 28 April 2026
Revised: 1 June 2026
Accepted: 4 June 2026
Published: 6 June 2026



## 1. Introduction

Distributed software development makes coordination both indispensable and difficult. Contributors must manage technical interdependence while operating across geographic, temporal, and organizational boundaries [1–3]. When that alignment fails, integration breakdowns, delayed delivery, and defects become more likely [2,4]. Socio-technical congruence (STC) theory has become a central explanation for this problem because it links software outcomes to the fit between technical needs and the coordination enacted by developers [5–7]. However, most STC research still rests, implicitly or explicitly, on a relatively static view of congruence. It largely treats alignment as a project-level condition that varies slowly over time. That assumption is increasingly inconsistent with the temporal structure of contemporary software development.

Software development does not typically proceed as a smooth, continuous flow. Across both industrial and open-source settings, development activity is episodic, marked by

periods of relative calm punctuated by bursts of intense commits, reviews, and contributor engagement [8,9]. These bursts frequently emerge around releases, urgent fixes, or major integration efforts, compressing coordination into narrow temporal windows [10]. In such episodes, technical dependencies become more immediate, contributor participation expands, and the cognitive and communicative demands placed on teams can outstrip routine coordination capacity. This creates a theoretical problem for existing STC research. If congruence is assumed to change only gradually, the theory risks overlooking the episodes in which coordination is under the greatest strain and most consequential for delivery.

This paper addresses that limitation by introducing development bursts—brief episodes of significant technical change and collaboration that alter coordination demands—as a theoretically meaningful unit of analysis for STC research. Drawing on STC theory [5], socio-technical systems theory [11,12], and research on coordination under pressure [13,14], it conceptualizes bursts as transient socio-technical stress events in which concentrated technical change can temporarily outpace the adaptive capacity of the coordination structure. This shift in temporal focus matters because it repositions congruence not as a background structural property but as an episodic and potentially fragile accomplishment.

Three gaps motivate the study. First, project-level treatments of congruence obscure short-term variation in coordination quality under changing workload conditions [5,15]. Second, prior work has not clearly distinguished burst intensity from coordination complexity. High activity volume does not necessarily imply architecturally complex change spanning interdependent modules and contributors [16,17]. Third, although prior research suggests that team familiarity and accumulated experience improve performance in demanding settings, their specific role in sustaining coordination quality during burst episodes remains insufficiently understood [14,18].

To address these gaps, the study develops and tests a burst-level model of coordination in distributed software projects. It examines how burst intensity and coordination complexity shape socio-technical congruence, how team experience conditions these relationships, and how burst conditions and congruence relate to delivery reliability. Empirically, the analysis draws on 45,981 development bursts from 6401 GitHub open-source projects and uses fixed-effects regression to isolate within-project variation.

The analysis yields three contributions. First, it extends STC theory beyond a predominantly static account of structural alignment and develops a more dynamic, episodic view of congruence. Second, it establishes development bursts as an analytically useful unit for examining coordination stress in repository data. Third, it shows at scale that the consequences of bursty work depend less on activity concentration alone than on structural coordination complexity and team capability. The broader implication is that coordination in distributed software development is not simply a matter of whether work intensifies, but of whether teams can sustain alignment when it does.

The remainder of the paper proceeds as follows. Section 2 develops the theoretical background and related work. Section 3 presents the conceptual model and hypotheses. Section 4 outlines the data, measures, and empirical strategy. Section 5 presents the results, while Section 6 discusses their implications and limitations and offers directions for future research. Section 7 concludes the paper.

## 2. Theoretical Background and Related Work

### *2.1. Development Bursts and the Temporal Concentration of Work*

Continuous software engineering rests on the premise that smaller batch sizes, automated integration, and shorter feedback loops improve software delivery [19,20]. However, in practice, software development rarely follows a smooth, continuous flow. Evidence from both industrial and open-source projects shows that development activity is episodic, with

periods of relative inactivity interrupted by bursts of intense commits, pull requests, and contributor engagement [8]. These episodes often emerge around release schedules, urgent fixes, or major integration events [21,22]. This temporal concentration matters because it reshapes the coordination problem that teams face. Aggregate repository measures can mask short-lived episodes in which coordination demands intensify, and delivery outcomes become most vulnerable.

At the same time, concentrated activity should not be treated as inherently dysfunctional. Under some conditions, bursts may reflect purposeful collective mobilization around salient integration demands, activating coordination rather than undermining it [13,23]. This makes one point clear: activity volume alone cannot explain the effects of bursts. What matters is not only how much work is concentrated into a short period, but also how that work is structured. A high-volume burst confined to a well-bounded component and handled by familiar contributors poses a different coordination problem from a moderate-volume burst that spans multiple subsystems and contributor groups [16,17]. Research on mining software repositories has not always clearly separated these dimensions. This study addresses that limitation by distinguishing activity concentration from structural coordination difficulty.

### *2.2. Socio-Technical Congruence*

STC theory builds on the broader socio-technical systems tradition, which holds that performance depends on the joint optimization of technical and social arrangements [12]. In software engineering, Conway's Law anticipated this logic by arguing that system architectures tend to mirror the communication structures of the organizations that produce them [24,25]. Cataldo et al. [5] provided the most influential empirical formulation of this idea by defining congruence as the alignment between software dependencies and the coordination activities of the developers who manage them. They showed that higher congruence was associated with shorter task completion times. Subsequent research has highlighted the importance of congruence in shaping software outcomes. Kwan et al. [6] showed that misalignment predicts build failures, while Cataldo and Herbsleb [4] linked gaps in congruence to decreased productivity and higher defect rates. Research has also shown that developers often lack awareness of the coordination requirements that arise from dependencies [26].

In large-scale agile settings, problems with alignment at the team-of-teams level often create significant integration challenges [27]. These dynamics become even more consequential in distributed software development. Geographic dispersion, asynchronous communication, organizational boundaries, and fluid participation weaken the informal coordination mechanisms on which teams often rely [4,28]. Open-source settings intensify these conditions because contributor roles remain fluid, and formal authority is limited [29,30]. Communication networks aligned with dependency structures lead to faster issue resolution and lower defect rates [31]. In contrast, architectural misalignment has been shown to predict higher technical debt [32].

Despite a substantial body of evidence, much STC research continues to treat congruence as a stable project-level characteristic. This design choice may obscure short-term fluctuations in coordination quality during periods of heightened activity [5,6]. Recent work similarly suggests that coordination quality is temporally sensitive and requires active management across different phases of work [27,33]. This study therefore conceptualizes congruence as a burst-level construct measured during concentrated episodes of development activity.

Mauerer et al. [34] demonstrated that alignment between socio-technical motifs does not exhibit a significant correlation with software bugs or project churn across 25 large open-

source projects. This indicates that, where congruence matters, its effects may manifest differently across levels of analysis. This does not invalidate socio-technical congruence theory. Rather, it suggests that the relationship between congruence and performance is more complex than earlier research has often implied. The present study addresses this issue by shifting the focus from project-level quality indicators to burst-level delivery reliability and by examining whether congruence during concentrated development reflects both alignment quality and coordination burden.

### *2.3. Repository-Based Proxies for Delivery Reliability*

The core question is how to assess delivery reliability when production-level operational metrics are unavailable. Delivery reliability refers to a team's ability to deliver work dependably in terms of timing, throughput, and quality, and it is shaped in part by the effectiveness of coordination [35]. In production environments, researchers commonly assess it using DevOps Research and Assessment metrics, including deployment frequency, lead time for changes, change failure rate, and mean time to recovery [35,36]. Open-source repositories rarely provide these indicators in a consistent form. As a result, researchers rely on repository-observable proxies such as issue resolution time, pull-request throughput, defect-related activity, and the ratio of closed to open issues [37,38]. These proxies are imperfect, but they remain analytically useful.

This distinction is important because delivery throughput and delivery reliability are related but not equivalent. Throughput refers to the volume or rate at which work items move through the repository, for example, issues closed or pull requests processed. Delivery reliability refers to the dependability of that flow: whether work is resolved within predictable timeframes, whether closure activity keeps pace with newly opened work, and whether short-term acceleration avoids unresolved coordination debt. This study therefore uses throughput-sensitive repository indicators as observable proxies for reliability, but interprets them cautiously as burst-level delivery responsiveness rather than as complete production-facing reliability measures.

Issue resolution time may capture both coordination efficiency and technical difficulty, while issue closure counts may reflect project-specific reporting practices rather than pure throughput [39,40]. Even so, they remain the standard basis for studying delivery outcomes in open-source settings, especially when the goal is to explain within-project variation rather than compare absolute performance across heterogeneous repositories [37,41].

Accordingly, this study uses three burst-level indicators of delivery reliability: average issue resolution time, issue closure count during the burst, and a coordination efficiency ratio based on resolved issues relative to opened issues. These measures are consistent with prior research on mining software repositories [42]. Resolution time is log-transformed to address positive skew [43], and a small number of negative values resulting from timestamp inconsistencies are set to zero prior to transformation, following established data-cleaning practices [16,39,41].

## 3. Conceptual Model and Hypothesis Development

This study addresses a core theoretical problem in distributed software development: when development activity becomes temporally concentrated, what determines whether teams sustain coordination or drift into misalignment? Existing research provides three complementary but only partially connected answers. STC theory explains why alignment between technical dependencies and enacted coordination matters for software outcomes [5]. Research on coordination under episodic pressure explains why concentrated work alters coordination demands rather than merely increasing workload [13,44]. Organizational learning research explains why experience can function as a capability that

allows teams to respond more effectively under demanding conditions [14,45]. Integrating these traditions suggests that development bursts should be understood as transient socio-technical episodes in which concentrated technical change raises coordination demands, but with consequences that depend on both the structure of the work and the capabilities available to manage it.

The framework rests on three linked components. The first is burst antecedents, captured by burst intensity and coordination complexity, which represent distinct forms of coordination demand. The second is team experience, which captures coordination capability through accumulated familiarity with the codebase, contributors, and routines. The third is socio-technical congruence, which functions both as an outcome of the interaction between burst demands and team capability and as a mechanism through which burst conditions may shape delivery reliability. Delivery reliability captures the extent to which teams sustain throughput and responsiveness during bursts. Figure 1 summarizes the model: burst intensity and coordination complexity serve as antecedents of congruence, team experience moderates these relationships, and congruence, together with the direct effects of burst conditions, is associated with burst-level delivery reliability.

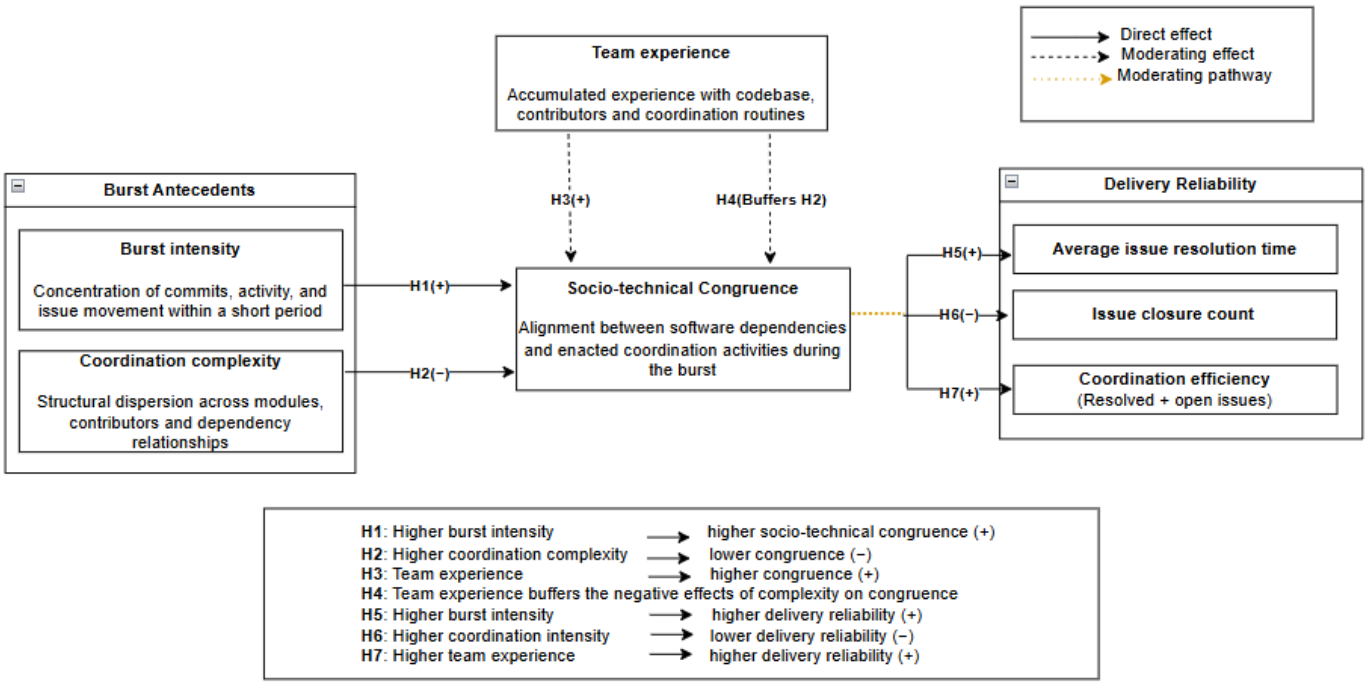


**Figure 1.** Conceptual model and hypotheses.

### *3.1. Burst Intensity and Congruence*

The first question is whether concentrated activity erodes congruence or mobilizes it. Burst intensity refers to the concentration of development activity within a bounded period, including code changes, contributor activity, and issue movement. As intensity increases, teams must manage dependencies, adjust to one another, and negotiate integration within narrower time windows [5,13]. One line of argument holds that congruence should weaken under these conditions. If required coordination expands faster than enacted coordination can adapt, alignment should deteriorate.

A competing line of argument points in the opposite direction. Research on coordination under pressure shows that salient integration demands can trigger compensatory communication and coordination effort rather than simple overload [13,23]. When actors recognize that interdependence has become more immediate, they may increase coordination around precisely those dependencies that matter most. In distributed software

development, intense bursts may therefore generate more enacted coordination around relevant technical ties rather than less. According to this logic, intensity encompasses not only the magnitude of pressure but also the degree of mobilization involved. Therefore, the following hypothesis is proposed:

**H1.** *Higher burst intensity is associated with higher socio-technical congruence.*

### *3.2. Coordination Complexity and Congruence*

The second question concerns the structure of the work rather than its volume. Coordination complexity is distinct from intensity. Intensity captures how much activity is concentrated in time; complexity captures how difficult it is to coordinate that activity across modules, contributors, and dependency relationships [16,17]. This distinction is theoretically necessary because not all bursts generate the same coordination problem. A burst can be intense without being structurally demanding, just as a burst of moderate volume can be difficult to coordinate if it spans multiple subsystems and contributor groups.

Organizational information-processing theory suggests that increasing complexity raises coordination demands in ways that existing communication channels may struggle to absorb, particularly in distributed settings marked by asynchronous interaction and fragmented awareness [46]. As architectural dispersion and contributor heterogeneity increase, the number of required coordination ties rises sharply [23]. In open-source settings, where participation is voluntary and formal authority is limited, the likelihood that all required ties will actually be enacted declines as the coordination burden grows [29,31]. The expected effect, therefore, is not simply more coordination but more missed coordination requirements. This suggests lower congruence, leading to the following hypothesis:

**H2.** *Higher coordination complexity is associated with lower socio-technical congruence.*

### *3.3. Team Experience as a Buffering Condition*

The third question is whether all teams face these burst conditions in the same way. Organizational learning theory suggests that they do not. Accumulated experience enables teams to build shared mental models, tacit coordination routines, and stronger mutual awareness, which in turn reduce cognitive overload and improve coordination under pressure [45,47]. In software development, prior collaboration and contributor tenure improve performance in complex and high-pressure settings [14,18,47]. At the project level, experience deepens familiarity with the codebase and its dependency structure [48]. At the team level, it enables faster mutual adjustment and more tacit forms of communication [49].

These arguments suggest two related expectations. First, experience should be positively associated with congruence because experienced teams are better able to recognize and enact required coordination ties. Second, its more consequential role should be conditional. Experience should matter most when coordination demands become structurally difficult, because familiarity with relevant components, contributors, and norms helps teams manage cross-cutting dependencies that would otherwise overwhelm routine coordination processes. In this sense, experience should buffer the adverse effect of coordination complexity on congruence. Accordingly, the following hypotheses are presented:

**H3.** *Team experience is positively associated with socio-technical congruence.*

**H4.** *Team experience attenuates the negative relationship between coordination complexity and socio-technical congruence.*

### *3.4. Burst Conditions, Congruence, and Delivery Reliability*

The final step in the model concerns delivery outcomes. STC theory holds that alignment improves software outcomes because developers responsible for interdependent work are more likely to coordinate in ways that prevent integration failures, reduce rework, and accelerate task completion [4,6]. This reasoning would suggest a positive relationship between congruence and delivery reliability. However, the burst context complicates that expectation.

Burst conditions may shape delivery reliability both directly and through congruence. Intense bursts may reflect productive mobilization that improves short-run throughput. By contrast, structurally complex bursts may generate integration burdens that reduce delivery reliability even when coordination effort is substantial. A further complication is that observed congruence during bursts may reflect not only alignment quality but also coordination burden. High observed coordination in a burst may indicate effective alignment, but it may also signal that teams are struggling to manage an unusually demanding dependency structure. For that reason, the downstream effect of congruence cannot be assumed to be straightforward.

The delivery component of the framework is therefore specified more cautiously than the antecedent models. The study expects burst intensity to improve delivery reliability, coordination complexity to reduce it, and team experience to improve it. By contrast, it examines the association between congruence and delivery reliability empirically rather than stating a strong directional hypothesis ex ante. The following hypotheses are proposed:

**H5.** *Higher burst intensity is associated with higher burst-level delivery reliability.*

**H6.** *Higher coordination complexity is associated with lower burst-level delivery reliability.*

**H7.** *Higher team experience is associated with higher burst-level delivery reliability.*

The framework advances STC research in two ways. First, it treats congruence as a dynamic burst-level condition rather than a static project property. Second, it distinguishes among activity mobilization, structural coordination difficulty, and accumulated coordination capability. That distinction provides a more precise basis for explaining when intense development episodes sustain delivery and when they undermine it.

## 4. Materials and Methods

### *4.1. The Dataset*

The empirical challenge in this study is to observe coordination under episodic pressure with sufficient granularity to capture meaningful variation across bursts of development. To address that challenge, the analysis uses a burst-level dataset comprising 45,981 development-burst observations drawn from 6401 open-source software projects hosted on GitHub. The dataset integrates repository event streams, including commit histories, pull request records, issue lifecycle traces, and contributor metadata from GitHub archives [50]. To focus on projects with sufficiently developed coordination structures, the sample includes only repositories that meet three minimum activity thresholds: at least 50 commits, at least 10 closed issues, and at least 24 months of observable activity. These criteria exclude inactive, archived, or trivially small repositories that are unlikely to yield meaningful evidence on coordination under episodic pressure [40].

The study identifies development bursts using a sliding-window algorithm applied to each project's commit-frequency time series. A burst is defined as a contiguous interval in which commit activity exceeds a project-specific threshold relative to the project's historical baseline. This operationalization aligns with the manuscript's conceptual definition of a

burst as a bounded episode of concentrated technical change. Project-specific normalization is necessary because baseline activity levels vary substantially across repositories. A single global threshold would confound episodic intensification with project scale and would systematically privilege larger or unusually active projects [8]. Each burst observation records the timing, duration, activity volume, coordination structure, and delivery outcomes of a single episode, producing a hierarchical dataset in which bursts are nested within projects.

### *4.2. Unit of Analysis*

The primary unit of analysis is the burst episode. This choice follows directly from the study's theoretical framing of bursts as transient socio-technical stress events in which coordination requirements intensify and teams must maintain alignment under compressed time constraints. Aggregating to the project or release level would obscure the within-project variation at the center of the argument, namely, how coordination alignment and delivery outcomes shift across episodes that differ in intensity, structural complexity, and experience composition [5,47]. The burst-level design also fits the estimation strategy. Because the analysis relies on within-project fixed effects, the substantive question is not whether more active projects differ from less active ones, but whether the same project exhibits different coordination and delivery patterns across bursts that vary in intensity, complexity, and team experience. On average, the final sample contains 7.2 burst observations per project, providing sufficient within-project variation to support fixed-effects estimation. Projects with fewer than three observed bursts remain in the full sample, although they contribute less to within-project identification. Therefore, the robustness checks report models that exclude these repositories alongside the main analyses.

### *4.3. Construct Operationalization*

The next step is to translate the theoretical constructs into repository-observable measures that capture burst conditions, coordination capability, congruence, and delivery outcomes. Burst intensity captures the concentration of development activity during a burst episode. It is measured as a standardized composite of three logged indicators: total activity events (commits and pull-request events), commit count, and lines of code changed. Log transformation reduces the influence of the strong positive skew typical of repository activity measures [8,43]. The analysis then standardizes and averages the resulting indicators to produce a burst-level intensity score centered on zero. This multi-indicator approach reduces dependence on any single activity measure that may be sensitive to project-specific recording conventions [5].

Coordination complexity captures the structural difficulty of the coordination problem associated with a burst rather than the sheer volume of activity involved. It is operationalized as a standardized composite of three logged indicators: file-module spread, total actor count, and committing-user count. These measures capture, respectively, the architectural dispersion of the changes, the breadth of the contributor set, and the number of contributors who directly execute code changes. Together, they represent the principal dimensions of structural coordination difficulty identified in prior software engineering research: architectural dispersion, actor-set size, and cross-component involvement [16,17].

Team experience represents the coordination capability available within a burst. It is measured as a standardized composite of two indicators: component experience, defined as the average number of prior commits made by active contributors to the components implicated in the burst, and contributor tenure, defined as the average elapsed time since each active contributor's first project contribution. These indicators capture both technical familiarity with the relevant codebase and social familiarity with project routines, coordination norms, and contributor networks [14,18,48].

Socio-technical congruence is measured using two indicators. Pull-request congruence (PR congruence) captures the extent to which developer pairs who share technical dependencies through pull-request activity also communicate during the burst window, thereby reflecting enacted coordination alignment during active integration work [5]. Structural congruence captures the alignment between formal component ownership and the contributor groups involved during the burst. Both measures range from 0 to 1. PR congruence serves as the primary measure of congruence in regression models because it exhibits greater within-sample variation, whereas structural congruence shows substantial ceiling effects. Structural congruence is retained for descriptive analysis and robustness checks.

Delivery reliability is operationalized as a burst-level composite of three standardized indicators: reverse-coded average issue resolution time, the number of issues closed during the burst window, and an efficiency ratio defined as the number of closed issues relative to the number of opened issues. Higher values indicate more favorable short-term delivery outcomes. This measure should be interpreted as a repository-observable proxy for delivery reliability rather than as a full DORA-style reliability construct, since production-facing indicators such as deployment failure and recovery time are not available in the present context [23,35]. Average resolution time is clipped at zero prior to log transformation to address a small number of negative values arising from timestamp inconsistencies in raw archival records [16,39].

### *4.4. Estimation Strategy*

The empirical question is whether changes in burst conditions within the same project are associated with changes in congruence and delivery reliability. To address this question, the analysis uses within-project fixed-effects estimation. Let b denote burst episodes and p denote projects. The general estimating equation is presented as Equation (1):

$$Y_{bp} = \alpha_p + X_{bp}\beta + \varepsilon_{bp} \quad (1)$$

In Equation (1), Ybp denotes the burst-level outcome for burst b in project p; αp captures project fixed effects; Xbp is the vector of burst-level predictors and controls; β is the vector of estimated coefficients; and εbp is the idiosyncratic error term. This specification removes time-invariant project-specific heterogeneity, including stable differences in architecture, governance, domain, contributor culture, and baseline activity level [51]. The coefficients are therefore identified entirely from within-project variation across bursts [52]. Standard errors are clustered at the project level to account for residual within-project dependence after the fixed-effects transformation [53]. All continuous variables are standardized prior to estimation.

The analysis estimates three models. Model 1 examines the antecedents of congruence, as specified in Equation (2):

$$\begin{aligned} PRCongruence_{bp} = {} & \alpha_p + \beta_1\, Intensity_{bp} + \beta_2\, Complexity_{bp} + \beta_3\, Experience_{bp} \\ & + \beta_4\, (Intensity \times Experience)_{bp} + \beta_5\, (Complexity \times Experience)_{bp} + \gamma C_{bp} + \varepsilon_{bp} \end{aligned} \quad (2)$$

In Equations (2)–(4), PRCongruencebp denotes pull-request congruence; DeliveryReliabilitybp denotes the burst-level delivery-reliability composite; Intensitybp, Complexitybp, and Experiencebp denote the standardized burst-level predictors; and Cbp is the control vector comprising burst duration, issue load, and team load. Interaction terms are entered as product terms after standardization.

Model 2 estimates the joint association of burst characteristics and congruence with delivery reliability, as specified in Equation (3):

$$DeliveryReliability_{bp} = \alpha_p + \beta_1\, Intensity_{bp} + \beta_2\, PRCongruence_{bp} + \beta_3\, Complexity_{bp} + \beta_4\, Experience_{bp} + \gamma C_{bp} + \varepsilon_{bp} \quad (3)$$

Model 3 extends Model 2 by adding the interaction between burst intensity and PR congruence, as specified in Equation (4):

$$DeliveryReliability_{bp} = \alpha_p + \beta_1\, Intensity_{bp} + \beta_2\, PRCongruence_{bp} + \beta_3\, Complexity_{bp} + \beta_4\, Experience_{bp} + \beta_5\, (Intensity \times PRCongruence)_{bp} + \gamma C_{bp} + \varepsilon_{bp} \quad (4)$$

The models examine three linked questions: how burst conditions shape congruence, how burst characteristics and congruence relate to delivery reliability, and whether the relationship between congruence and delivery reliability changes under more intense episodic conditions.

The fixed-effects design strengthens inference relative to cross-sectional models by isolating within-project variation rather than conflating it with stable differences across projects. Even so, the design does not eliminate within-burst simultaneity. Congruence and delivery outcomes may still be jointly shaped during the same episode, particularly when coordination activity responds endogenously to emerging delivery pressures. The estimates should therefore be interpreted as robust within-project associations rather than as strictly identified causal effects [51,52]. More demanding identification strategies, including temporally lagged congruence measures or instrumental-variable designs, remain promising directions for future research.

### *4.5. Robustness, Endogeneity, and Diagnostic Strategy*

Several additional checks were added to address construct validation, simultaneity, and interpretation of the congruence coefficients. First, the study evaluates operationalization sensitivity by replacing the composite burst-intensity and coordination-complexity measures with their component indicators and by estimating stricter burst-inclusion specifications. These checks assess whether the main conclusions are driven by a particular composite construction or by lower-intensity episodes. Second, lagged congruence specifications separate prior burst-level congruence from contemporaneous delivery outcomes, thereby reducing but not eliminating simultaneity concerns. Third, residualized PR congruence models examine whether the sign reversal between the positive bivariate correlation and negative conditional coefficient reflects simple multicollinearity or a more substantive distinction between alignment quality and coordination burden. Fourth, experience-quartile diagnostics test the ceiling-effect interpretation of the negative burst-intensity by experience interaction. Finally, the non-significant interaction between burst intensity and PR congruence is interpreted using confidence intervals and a practical equivalence bound rather than significance testing alone.

These analyses are diagnostic rather than fully causal. The available repository data still do not provide exogenous shocks to congruence, and some coordination may occur outside observable GitHub channels. Consequently, the revised analysis treats fixed-effects coefficients as robust within-project associations and reserves causal language for theoretically justified mechanisms rather than statistical identification alone.

## 5. Results

### *5.1. Descriptive Statistics and Bivariate Associations*

The first question is how burst episodes are distributed across the sample and how the key constructs relate to one another before multivariate estimation. Table 1 reports descriptive statistics for the principal burst-level variables across the analytical sample of 45,981 bursts from 6401 projects.

**Table 1.** Descriptive statistics for burst-level variables.

| Variable | Mean | Median | SD | Min | Max |
|---|---|---|---|---|---|
| Burst intensity (standardized) | 0.000 | −0.404 | 2.616 | −5.332 | 19.176 |
| Coordination complexity (standardized) | 0.000 | −0.491 | 2.410 | −4.584 | 37.004 |
| Team experience (standardized) | 0.000 | 0.038 | 1.654 | −5.661 | 6.590 |
| PR congruence | 0.603 | 0.667 | 0.412 | 0.000 | 1.000 |
| Structural congruence | 0.802 | 1.000 | 0.360 | 0.000 | 1.000 |
| Delivery reliability (composite) | 0.000 | −0.165 | 1.630 | −4.896 | 16.228 |
| Average resolution time (hours, clipped) | 20.592 | 11.145 | 43.316 | 0.000 | 2913.470 |
| Issues closed | 15.729 | 4.000 | 271.854 | 0.000 | 41,199.000 |
| Efficiency ratio | 2.591 | 1.500 | 5.169 | 0.000 | 558.000 |

Note: Burst intensity, coordination complexity, team experience, and delivery reliability are summed composites of log-transformed, z-standardized indicators. A value of zero denotes the sample mean of the transformed component indicators, not the absence of the construct. Coefficients for these predictors should be interpreted as changes associated with a one-standard-deviation increase in the transformed component scale. Average resolution time is clipped at zero before log transformation to address timestamp inconsistencies in raw archival records. All multivariate models incorporate project-level fixed effects.

Two distributional features stand out. First, both burst intensity and coordination complexity are strongly right-skewed. In each case, the mean exceeds the median, and the upper tail extends well beyond the central mass of observations. This pattern is consistent with prior research on mining software repositories, which shows that development activity is episodic rather than smoothly continuous: most bursts are moderate in scale, while a smaller number are exceptionally intense or structurally demanding [8,9]. Second, the two congruence measures move together but remain clearly distinct. PR congruence shows substantially greater within-sample variation than structural congruence, whereas structural congruence clusters heavily at its upper bound. This ceiling effect supports using PR congruence as the principal alignment measure in the regression models. Table 2 reports the Spearman correlation matrix.

**Table 2.** Spearman correlation matrix.

| Variable | (1) | (2) | (3) | (4) | (5) |
|---|---|---|---|---|---|
| (1) Burst intensity | 1.000 | | | | |
| (2) Coordination complexity | 0.580 | 1.000 | | | |
| (3) Team experience | −0.025 | 0.004 | 1.000 | | |
| (4) Pull-request congruence | 0.100 | −0.132 | 0.066 | 1.000 | |
| (5) Delivery reliability | 0.401 | 0.215 | −0.012 | 0.113 | 1.000 |

Note: All coefficients are Spearman rank correlations computed using burst-level variables. These descriptive statistics do not account for within-project covariation.

Two descriptive patterns are especially informative. Burst intensity is positively correlated with delivery reliability, suggesting that, at the bivariate level, concentrated activity is associated with stronger short-run throughput rather than with immediate deterioration in delivery. Coordination complexity is also strongly and positively correlated with burst intensity, confirming that intense bursts are often structurally demanding. This covariation makes multivariate estimation necessary, because it is not possible to infer from descriptive correlations alone whether activity concentration and structural coordination difficulty have distinct effects. By contrast, the near-zero raw association between team experience and both intensity and complexity supports treating it as a separate coordination capability rather than a proxy for project busyness or scale.

### *5.2. Antecedents of Socio-Technical Congruence During Bursts*

The next question is what drives socio-technical congruence during burst episodes. Table 3 reports the fixed-effects model predicting pull-request congruence from burst intensity, coordination complexity, team experience, and their interactions, while controlling for burst duration, issue load, and team load. The within-project $R^2$ of 0.111 indicates that the model explains a meaningful share of within-project variation in congruence, especially given the heterogeneity of open-source burst episodes and the stringency of the fixed-effects specification.

**Table 3.** Fixed-effects model predicting pull-request congruence.

| Variable | Coefficient (Clustered SE) |
|---|---|
| Burst intensity | 0.056 *** (0.001) |
| Coordination complexity | −0.005 ** (0.002) |
| Team experience | 0.036 *** (0.001) |
| Burst intensity × Team experience | −0.003 *** (0.001) |
| Coordination complexity × Team experience | 0.004 *** (0.001) |
| Burst duration | −0.063 *** (0.004) |
| Issue load | −0.168 *** (0.008) |
| Team load | 0.192 *** (0.006) |
| Within-project $R^2$ | 0.111 |

Note: Project fixed effects are included. Cluster-robust standard errors are reported in parentheses. *** $p < 0.001$; ** $p < 0.01$.

The model yields a clear result. Burst intensity is positively and statistically significantly associated with PR congruence ($\beta = 0.056$, $p < 0.001$), supporting H1. Within projects, more intense bursts are associated with higher, not lower, levels of observed coordination alignment. This pattern suggests that concentrated work episodes often mobilize coordination rather than suppress it, which is consistent with the view that intense bursts can trigger compensatory communication and review effort in distributed development [23].

Coordination complexity shows the opposite pattern. It is negatively associated with PR congruence ($\beta = -0.005$, $p < 0.01$), supporting H2. Bursts with greater architectural dispersion and broader actor involvement exhibit lower alignment, even after accounting for activity concentration and load conditions. This finding accords with information-processing arguments that structurally difficult coordination problems place demands on communication systems that are especially difficult to satisfy in distributed, asynchronous settings [46].

Team experience has a positive direct effect on congruence ($\beta = 0.036$, $p < 0.001$), supporting H3. This result aligns with organizational learning arguments that accumulated familiarity with code, routines, and collaborators improves coordination under pressure [14,45]. The interaction terms sharpen that interpretation. The interaction between burst intensity and team experience is negative and statistically significant ($\beta = -0.003$, $p < 0.001$). This does not reverse the positive main effects of intensity and experience. Instead, it indicates that the positive association between intensity and congruence is slightly weaker in more experienced contexts. One plausible interpretation is a ceiling effect: experienced teams may already sustain relatively high congruence at moderate intensity levels, leaving less scope for further mobilization-driven gains as activity increases.

By contrast, the interaction between coordination complexity and team experience is positive and statistically significant ($\beta = 0.004$, $p < 0.001$), supporting H4. Experience attenuates the negative effect of structural coordination difficulty on congruence. This pattern is consistent with the argument that accumulated project and team familiarity improves the capacity to navigate cross-cutting dependencies [14,54].

These findings show that intensity and complexity capture different dimensions of burst pressure. Intensity is associated with greater coordination alignment, whereas complexity reduces it. Experience improves congruence directly and, more significantly, buffers the adverse effect of structural coordination difficulty.

### *5.3. Burst Characteristics, Congruence, and Delivery Reliability*

The final step is to examine how burst conditions and congruence relate to delivery reliability. Table 4 reports on the baseline fixed-effects model predicting burst-level delivery reliability.

**Table 4.** Fixed-effects model predicting burst-level delivery reliability.

| Variable | Coefficient (Clustered SE) |
|---|---|
| Burst intensity | 0.091 *** (0.003) |
| PR congruence | −0.137 *** (0.011) |
| Coordination complexity | −0.029 *** (0.005) |
| Team experience | 0.013 *** (0.003) |
| Burst duration | −0.466 *** (0.009) |
| Issue load | 0.776 *** (0.020) |
| Team load | 0.876 *** (0.014) |
| Within-project $R^2$ | 0.687 |

Note: Project fixed effects are included. Cluster-robust standard errors are reported in parentheses. *** $p < 0.001$.

This specification explains a substantial share of within-project variation in delivery reliability, with a within-project $R^2$ of 0.687. Issue load and team load are the largest predictors, which is consistent with the throughput-sensitive nature of the composite delivery measure.

Burst intensity is positively associated with delivery reliability ($\beta = 0.091$, $p < 0.001$), supporting H5. Within projects, more intense bursts are associated with greater throughput, faster issue movement, and higher closure efficiency. This result matches the descriptive pattern in Table 2 and supports the productive-mobilization interpretation: many bursts appear to be concentrated but effective episodes of collective work rather than instances of dysfunctional overload.

Coordination complexity again works in the opposite direction. It is negatively associated with delivery reliability ($\beta = -0.029$, $p < 0.001$), supporting H6. Even after controlling for intensity, bursts that are more architecturally dispersed and cross-cutting reduce throughput and responsiveness. This finding is consistent with prior evidence that architectural dispersion and cross-component coordination increase integration cost and rework [16,17].

Team experience is positively associated with delivery reliability ($\beta = 0.013$, $p < 0.001$), supporting H7. Although this effect is smaller than those for issue load or team load, it remains theoretically meaningful because it reinforces the view of experience as a coordination capability rather than a simple correlate of project maturity [18,45].

The most theoretically challenging result is the negative coefficient on PR congruence ($\beta = -0.137$, $p < 0.001$). The revised conceptual model treats the congruence–delivery relationship as empirically open rather than as a strong directional hypothesis, so this result requires careful interpretation rather than simple disconfirmation. Once the model includes demand-side controls, higher observed congruence may partly reflect bursts in which dependency-management demands are unusually salient. On that reading, coordination is activated precisely because the episode is difficult. Observed congruence during bursts may therefore proxy not only coordination quality but also coordination burden. This interpretation is consistent with arguments that high levels of observed

coordination can sometimes signal the necessity of coordination rather than frictionless collaboration [5,55]. The sign reversal between the positive raw correlation in Table 2 and the negative conditional coefficient in Table 4 is also consistent with suppressor dynamics in multivariate models [56].

Table 5 adds an interaction between burst intensity and PR congruence as an exploratory test of whether the association between congruence and delivery reliability changes across more- and less-intense bursts.

**Table 5.** Fixed-effects moderation model predicting burst-level delivery reliability.

| Variable | Coefficient (Clustered SE) |
|---|---|
| Burst intensity | 0.092 *** (0.003) |
| PR congruence | −0.142 *** (0.012) |
| Burst intensity × PR congruence | −0.007 (0.005) |
| Coordination complexity | −0.029 *** (0.005) |
| Team experience | 0.013 *** (0.003) |
| Burst duration | −0.465 *** (0.009) |
| Issue load | 0.773 *** (0.020) |
| Team load | 0.878 *** (0.014) |
| Within-project $R^2$ | 0.687 |

Note: Project fixed effects are included. Cluster-robust standard errors are reported in parentheses. *** $p < 0.001$.

The interaction between burst intensity and PR congruence is not statistically significant ($\beta = -0.007$, $p > 0.05$). This suggests that, once the model includes the main effects and controls, the conditional association between congruence and delivery reliability does not vary systematically with burst intensity. The negligible change in within-project $R^2$ leads to the same conclusion: the moderation term does not materially improve model fit. Reporting this null result is analytically important because it avoids attributing a conditional structure to the data where the evidence does not support it [57].

### *5.4. Additional Robustness and Diagnostic Analyses*

Table 6 reports sensitivity checks for the principal delivery-reliability model. The positive association between burst intensity and delivery reliability remains stable across composite, commit-only, activity-only, and lines-of-code intensity specifications. Coordination complexity remains negative in the baseline, module-spread, and actor-set specifications, although the effect attenuates in the most restrictive high-intensity subsamples because restricting the range of burst intensity also removes part of the variation through which structural complexity operates. The negative conditional coefficient on PR congruence is also stable across the main sensitivity checks, supporting the interpretation that contemporaneous PR congruence partly captures coordination burden during demanding bursts rather than pure alignment quality.

To address simultaneity more directly, Table 7 reports lagged and residualized congruence checks. Lagged PR congruence does not significantly predict current delivery reliability, and current PR congruence does not predict next-burst delivery reliability after fixed effects and controls. This weakens any strong causal interpretation of the contemporaneous negative coefficient. The residualized PR congruence model remains negative, indicating that the sign reversal is not merely a mechanical artifact of bivariate correlation or simple collinearity. Instead, the evidence supports a more cautious interpretation: observed PR congruence during bursts combines dependency-aligned coordination with reactive coordination burden.

**Table 6.** Sensitivity checks for the delivery-reliability model.

| Specification | N | Intensity Coefficient | Complexity Coefficient | PR Congruence Coefficient | Interpretation |
|---|---|---|---|---|---|
| Baseline composite | 45,981 | 0.091 *** | −0.029 *** | −0.137 *** | Main pattern retained |
| Stricter burst filter: upper 75% by intensity | 34,486 | 0.112 *** | −0.018 * | −0.155 *** | Direction retained |
| Stricter burst filter: upper 50% by intensity | 22,991 | 0.123 *** | −0.002 n.s. | −0.203 *** | Complexity attenuates under range restriction |
| Commit-only intensity | 45,981 | 0.129 *** | −0.019 ** | −0.098 *** | Intensity robust |
| Activity-only intensity | 45,981 | 0.084 *** | −0.014 † | −0.091 *** | Direction retained |
| Module-spread complexity | 45,981 | 0.094 *** | −0.048 *** | −0.129 *** | Structural risk retained |
| Actor-set complexity | 45,981 | 0.087 *** | −0.078 *** | −0.145 *** | Actor dispersion risk retained |

Note: Project fixed effects and clustered standard errors are retained across models. n.s. = not significant; † $p < 0.10$; * $p < 0.05$; ** $p < 0.01$; *** $p < 0.001$. The stricter burst filters approximate alternative burst-inclusion thresholds using the processed burst-level dataset.

**Table 7.** Lagged, residualized, and equivalence diagnostics.

| Diagnostic Check | N | Key Estimate | Inference | Interpretation |
|---|---|---|---|---|
| Lagged PR congruence – current reliability | 39,580 | β = 0.018 (SE = 0.013) | n.s. | No strong lagged reliability effect |
| Lagged structural congruence – current reliability | 39,580 | β = 0.007 (SE = 0.015) | n.s. | Structural congruence is not predictive once lagged |
| Current PR congruence – next-burst reliability | 39,580 | β = 0.004 (SE = 0.023) | n.s. | No evidence of forward predictive effect |
| Residualized PR congruence | 45,981 | β = −0.137 (SE = 0.014) | $p < 0.001$ | Negative coefficient is not only a collinearity artifact |
| Burst intensity × PR congruence equivalence check | 45,981 | β = −0.007; 95% CI [−0.017, 0.003] | within ±0.02 bound | Interaction is practically negligible |

Note: Lagged models order bursts temporally within projects. n.s. = not significant. The equivalence bound is defined as ±0.02 standardized units, a deliberately conservative threshold for a substantively negligible interaction effect.

Table 8 provides descriptive evidence for the ceiling-effect interpretation of the burst-intensity by team-experience interaction. High-experience bursts start from a higher average level of PR congruence than low-experience bursts. The estimated intensity slope is smaller in the highest experience quartile than in the lowest experience quartile, suggesting that experienced teams have less remaining scope for mobilization-driven increases in observed congruence. This evidence supports the ceiling-effect interpretation, but the revised manuscript treats it as descriptive rather than definitive.

**Table 8.** PR congruence by experience quartile and burst-intensity level.

| Experience Quartile | Mean PR Congruence | Low Intensity | Moderate Intensity | High Intensity | Intensity Slope |
|---|---|---|---|---|---|
| Q1 (low) | 0.565 | 0.456 | 0.613 | 0.626 | 0.065 *** |
| Q2 | 0.596 | 0.544 | 0.641 | 0.602 | 0.049 *** |
| Q3 | 0.614 | 0.540 | 0.652 | 0.647 | 0.051 *** |
| Q4 (high) | 0.638 | 0.550 | 0.653 | 0.709 | 0.049 *** |

Note: Low, moderate, and high intensity are within-quartile terciles of burst intensity. Slopes are fixed-effects estimates of PR congruence on burst intensity within experience quartiles, controlling for complexity, duration, issue load, and team load. *** $p < 0.001$.

The results support a differentiated account of burst dynamics consistent with the revised model. H1–H4 are supported: higher burst intensity is linked to congruence; coordination complexity reduces congruence; team experience enhances congruence; and experience lessens the negative impact of complexity. H5–H7 are also supported: burst intensity and team experience are positively associated with delivery reliability, whereas coordination complexity is negatively associated with it. The one conceptually difficult result is the negative conditional coefficient on PR congruence in the delivery model. Rather than undermining the framework, this result suggests that the empirical meaning of observed congruence under burst conditions is more complex than standard STC formulations allow, because congruence may capture both alignment quality and coordination burden. The broader implication is clear: the risks of development bursts appear to be primarily structural rather than merely volumetric.

## 6. Discussion

The central question in this study is whether development bursts primarily disrupt coordination or instead mobilize it. The results point clearly toward the latter. Within projects, burst intensity is positively associated with both socio-technical congruence and delivery reliability. This finding challenges the common assumption that concentrated development activity is inherently pathological or that temporal concentration necessarily produces coordination overload and degraded performance [58,59]. Instead, the evidence suggests that many burst episodes represent purposeful collective responses to salient, time-bounded integration demands. As projects move toward release milestones, respond to urgent defects, or implement dependency upgrades, contributors appear to intensify not only their work but also their coordination. This interpretation is consistent with research showing that teams under pressure can activate latent coordination capacity when task urgency becomes salient [13,23,44]. Development bursts are therefore better understood as a distinct coordination regime than as simple departures from a stable development baseline.

At the same time, the results identify a different and more consistent source of degraded outcomes: coordination complexity. Bursts that are more architecturally dispersed, involve broader sets of actors, and span cross-cutting dependencies are associated with both lower congruence and lower delivery reliability. This distinction matters theoretically because software engineering research has often treated intense activity and difficult coordination as closely coupled, even though activity volume and structural coordination difficulty are analytically distinct [16,17]. The pattern fits organizational information-processing theory. Structurally complex bursts increase the number of coordination ties that teams must enact and widen the range of knowledge they must integrate, thereby creating demands that existing communication channels cannot easily absorb. These pressures become even stronger in distributed and asynchronous settings, where awareness is

fragmented and coordination repair is slower [28,46]. The main risks of bursts are therefore structural rather than volumetric.

The findings also refine the role of team experience. Experience has a positive direct association with congruence and a modest positive direct association with delivery reliability, but its more consequential role is conditional: it attenuates the negative relationship between coordination complexity and congruence. This result supports the organizational learning argument that accumulated experience builds coordination capability through shared mental models, tacit routines, and familiarity with both the codebase and the contributor network [14,45,60]. The asymmetry in the moderation effects is especially revealing. Experience does not buffer burst intensity as hypothesized. Instead, the positive intensity–congruence association is slightly weaker in more experienced contexts, likely because these teams already maintain relatively high alignment at moderate intensity levels. By contrast, experience matters clearly when bursts are structurally difficult, because familiarity with components, contributors, and coordination norms helps teams manage dependency-related demands more effectively [18,54]. Experience thus operates less as a general productivity premium than as a targeted coordination capability whose value rises under structural pressure.

The most theoretically difficult result is the negative conditional association between pull-request congruence and delivery reliability after controlling for issue load and team load. This result should not be read as evidence that alignment is harmful, since that conclusion would contradict the core STC literature linking higher congruence to faster completion, fewer failures, and lower defect rates [4,6]. A more plausible interpretation is that, under burst conditions, observed congruence captures not only alignment quality but also coordination burden. In difficult bursts, contributors may communicate more precisely because the integration problem itself is demanding. Under those conditions, higher observed congruence may coexist with lower-than-expected throughput once demand-side predictors are held constant. Congruence, therefore, becomes partly endogenous to the coordination problem that also depresses delivery performance. The sign reversal between the raw positive association and the negative conditional coefficient is consistent with suppressor dynamics and reverse-causal contamination in burst-level fixed-effects models [52,56]. This result identifies an important boundary condition for applying standard STC measures to high-frequency repository data.

### *6.1. Theoretical and Practical Implications*

These findings carry both theoretical and practical implications. Theoretically, they extend STC research in four ways. First, they support a more temporally sensitive account of congruence. Congruence is not simply a slowly varying structural property of a project; it is an episodic state that fluctuates meaningfully across bursts. Second, burst intensity and coordination complexity should not be treated as interchangeable forms of pressure. Intensity often mobilizes coordination, whereas complexity degrades it. Third, team experience matters primarily as a capability to absorb structural coordination difficulties rather than as a general productivity premium. Fourth, the empirical meaning of congruence under burst conditions is more complex than conventional STC formulations assume, because observable coordination may partly proxy coordination burden rather than pure alignment quality. These findings move the STC research program beyond a static structural account toward a more episodic and contingent one, better aligned with the uneven temporal rhythms of contemporary distributed software development.

The findings also position burst-level STC within a broader literature on adaptive and reconfigurable systems. Recent work on bedrock models in communication and sensing, for example, emphasizes modularity, transferability, and robustness under changing system

conditions [61]. Although that study addresses a different technical domain, the conceptual parallel is useful: robust performance in dynamic systems depends not merely on local optimization, but on whether system components can be recombined or reconfigured without losing functional alignment. In the present study, congruence plays an analogous socio-technical role by indicating whether the coordination structure can adapt quickly enough to match changing technical dependencies during bursts.

The practical implications are equally clear. Managers should not treat all bursts as harmful or attempt to eliminate concentrated activity as such. Because burst intensity is positively associated with both congruence and delivery reliability, suppressing all bursts may reduce beneficial mobilization without improving outcomes. The more pressing governance problem is the structural complexity of coordination. Efforts to improve delivery under burst conditions should therefore focus on architectural modularity, clearer dependency boundaries, contributor continuity, and knowledge transfer. Bursts that span many modules and involve loosely familiar contributors should be treated as higher-risk episodes requiring stronger coordination support than bursts that are merely intense. Tooling and analytics should likewise focus less on raw activity spikes and more on structural warning signals such as rising cross-module dispersion, expanding actor sets, and unfamiliar contributor combinations. In distributed settings, the priority is not to eliminate bursts, but to identify which bursts are most likely to exceed the project's current coordination capacity.

### *6.2. Threats to Validity*

Several validity concerns warrant attention. First, construct validity depends on operationalizing pull-request congruence as the primary measure of alignment. Structural congruence was retained for comparison, but its ceiling effects make it less informative at the burst level. Delivery reliability is also measured through repository-observable proxies rather than production telemetry, so the dependent variable captures short-run issue flow and responsiveness rather than the full DORA construct [35,39].

Second, internal validity remains constrained by simultaneity within bursts. Project fixed effects absorb time-invariant heterogeneity, but they do not eliminate reverse causation when coordination activity increases in response to difficult episodes. This limitation is especially relevant for the negative conditional coefficient on pull-request congruence. Third, external validity is limited by the GitHub-based open-source setting, where voluntary participation, fluid roles, and weak formal authority differ from proprietary environments [29]. Finally, reliability depends on the transparency of the analytical pipeline. To support replication, future research and replication materials should document all operationalization decisions, preprocessing steps, and estimation code.

### *6.3. Limitations and Future Research Avenues*

Several limitations should shape how these findings are interpreted. First, the study relies on observable repository proxies to assess socio-technical congruence and delivery reliability. Although such measures are standard in mining software research, they do not fully capture factors such as private communication or deployment challenges. The findings should therefore be interpreted as evidence about burst-level coordination and short-term performance rather than as a complete account of operational reliability.

Second, the fixed-effects design effectively removes time-invariant project differences, but it does not fully address simultaneity within bursts, in which coordination and delivery outcomes may evolve together. This issue is especially relevant to the negative relationship between PR congruence and delivery reliability, which likely reflects both alignment quality and coordination burden. The lagged and residualized diagnostics added in the

revision reduce the risk of over-interpretation but do not convert the design into a causal identification strategy. The estimates therefore capture robust within-project associations rather than strictly causal effects.

Third, the analysis depends on specific operational choices, including the use of composite indicators for burst intensity and coordination complexity. Alternative measures could generate different estimates. Pull-request congruence serves as the primary alignment metric because it provides greater within-sample variation, but other indicators may capture different dimensions of socio-technical fit. Fourth, the sample is limited to GitHub-hosted open-source projects. That setting offers strong visibility into distributed coordination, but it also limits external validity because open-source projects differ from proprietary environments in governance, incentives, and communication norms.

These limitations point to several avenues for future research. Stronger causal designs, including temporally lagged congruence measures, event-study approaches, or instrumental-variable strategies, would help more clearly separate alignment quality from the reactive coordination burden. Richer data that combine repository traces with communication logs, deployment telemetry, or incident records could be used to test whether the observed burst-level patterns extend to operational reliability and post-release quality. Replication in industrial settings would further clarify the boundary conditions of the findings. Qualitative and mixed-methods research could also deepen the explanation by showing how teams mobilize and coordinate during burst episodes.

## 7. Conclusions

This study examined how development bursts, defined as concentrated episodes of technical change and collaboration, shape socio-technical congruence and delivery reliability in distributed open-source software projects. Using 45,981 bursts from 6401 projects and within-project fixed-effects models, it makes four principal contributions. First, development bursts are not uniformly disruptive. Higher burst intensity is positively associated with both congruence and delivery reliability, which indicates that many bursts reflect productive mobilization around shared integration demands rather than breakdowns in coordination. Second, the main risks of bursts are structural rather than volumetric. Coordination complexity consistently reduces both congruence and delivery reliability, directing attention to architectural dispersion, cross-component dependencies, and heterogeneous contributor involvement as the main sources of burst fragility. Third, team experience functions as a targeted coordination capability. It improves alignment directly and buffers against the negative effects of coordination complexity, highlighting the importance of contributor continuity, familiarity, and knowledge transfer in sustaining resilience under episodic stress. Finally, the study identifies an important measurement challenge for burst-level STC research: observed congruence may capture both alignment quality and coordination burden, creating endogeneity that complicates causal interpretation.

The broader implication is that socio-technical coordination in software development is episodic, contingent, and structurally conditioned. Bursts are not merely deviations from a smooth workflow; they are coordination regimes whose consequences depend on the architecture of the work and the capabilities of the team. Future research should therefore strengthen temporal separation between congruence measurement and delivery outcomes, improve identification strategies, and replicate these patterns in industrial settings with richer performance telemetry.

**Funding:** This research received no external funding.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** The dataset used in this study was derived from the publicly available Modeling Productivity in Open Source GitHub Projects: A Dataset and Codebase, deposited by Choudhary, Bogart, Rose and Herbsleb at Carnegie Mellon University's KiltHub/Figshare repository (DOI: 10.1184/R1/6397013). The associated codebase is available through the GitHub repository samridhishree/GitHub_Productivity.

**Conflicts of Interest:** The author declares no conflicts of interest.

## Abbreviations

The following abbreviations are used in this manuscript:

| | |
|---|---|
| STC | Socio-technical congruence |
| DORA | DevOps Research and Assessment |
| PR | Pull-request congruence |

## References

1. Espinosa, J.A.; Slaughter, S.A.; Kraut, R.E.; Herbsleb, J.D. Team Knowledge and Coordination in Geographically Distributed Software Development. *J. Manag. Inf. Syst.* **2007**, *24*, 135–169. [CrossRef]
2. Herbsleb, J. Building a Socio-Technical Theory of Coordination: Why and How (Outstanding Research Award). In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*; Association for Computing Machinery: New York, NY, USA, 2016; pp. 2–10.
3. Kiely, G.; Butler, T.; Finnegan, P. Global Virtual Teams Coordination Mechanisms: Building Theory from Research in Software Development. *Behav. Inf. Technol.* **2022**, *41*, 1952–1972. [CrossRef]
4. Cataldo, M.; Herbsleb, J.D. Coordination Breakdowns and Their Impact on Development Productivity and Software Failures. *IEEE Trans. Softw. Eng.* **2013**, *39*, 343–360. [CrossRef]
5. Cataldo, M.; Herbsleb, J.D.; Carley, K.M. Socio-Technical Congruence: A Framework for Assessing the Impact of Technical and Work Dependencies on Software Development Productivity. In *Proceedings of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*; Association for Computing Machinery: New York, NY, USA, 2008; pp. 2–11.
6. Kwan, I.; Schroter, A.; Damian, D. Does Socio-Technical Congruence Have an Effect on Software Build Success? A Study of Coordination in a Software Project. *IEEE Trans. Softw. Eng.* **2011**, *37*, 307–324. [CrossRef]
7. Namal Rajapakse, R.; Szabo, C. Towards Multi-Class Socio-Technical Congruence: Assessing Coordination in Collaborative Software Development Settings. *J. Softw. Evol. Process* **2025**, *37*, e70040. [CrossRef]
8. Claes, M.; Mäntylä, M.V.; Kuutila, M.; Adams, B. Do Programmers Work at Night or During the Weekend? In *Proceedings of the 40th International Conference on Software Engineering*; ACM: Gothenburg, Sweden, 2018; pp. 705–715.
9. Choudhary, S.; Bogart, C.; Rose, C.; Herbsleb, J. Using Productive Collaboration Bursts to Analyze Open Source Collaboration Effectiveness. In Proceedings of the 2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER), London, ON, Canada, 18–21 February 2020; pp. 400–410.
10. Butler, S.; Gamalielsson, J.; Lundell, B.; Brax, C.; Mattsson, A.; Gustavsson, T.; Feist, J.; Kvarnström, B.; Lönroth, E. Considerations and Challenges for the Adoption of Open Source Components in Software-Intensive Businesses. *J. Syst. Softw.* **2022**, *186*, 111152. [CrossRef]
11. Trist, E.L.; Bamforth, K.W. Some Social and Psychological Consequences of the Longwall Method of Coal-Getting. *Hum. Relat.* **1951**, *4*, 3–38. [CrossRef]
12. Baxter, G.; Sommerville, I. Socio-Technical Systems: From Design Methods to Systems Engineering. *Interact. Comput.* **2011**, *23*, 4–17. [CrossRef]
13. Faraj, S.; Xiao, Y. Coordination in Fast-Response Organizations. *Manag. Sci.* **2006**, *52*, 1155–1169. [CrossRef]
14. Huckman, R.S.; Staats, B.R. Fluid Tasks and Fluid Teams: The Impact of Diversity in Experience and Team Familiarity on Team Performance. *Manuf. Serv. Oper. Manag.* **2011**, *13*, 310–328. [CrossRef]
15. Dingsøyr, T.; Nerur, S.; Balijepally, V.; Moe, N.B. A Decade of Agile Methodologies: Towards Explaining Agile Software Development. *J. Syst. Softw.* **2012**, *85*, 1213–1221. [CrossRef]
16. Bird, C.; Nagappan, N.; Gall, H.; Murphy, B.; Devanbu, P. Putting It All Together: Using Socio-Technical Networks to Predict Failures. In Proceedings of the 2009 20th International Symposium on Software Reliability Engineering, Mysuru, India, 16–19 November 2009; pp. 109–119.
17. MacCormack, A.; Baldwin, C.; Rusnak, J. Exploring the Duality Between Product and Organizational Architectures: A Test of the "Mirroring" Hypothesis. *Res. Policy* **2012**, *41*, 1309–1324. [CrossRef]

18. Foucault, M.; Palyart, M.; Blanc, X.; Murphy, G.C.; Falleri, J.-R. Impact of Developer Turnover on Quality in Open-Source Software. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*; Association for Computing Machinery: New York, NY, USA, 2015; pp. 829–841.
19. Humble, J.; Farley, D. *Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation*; Pearson Education: London, UK, 2010.
20. Fitzgerald, B.; Stol, K.-J. Continuous Software Engineering: A Roadmap and Agenda. *J. Syst. Softw.* **2017**, *123*, 176–189. [CrossRef]
21. Adams, B.; Bellomo, S.; Bird, C.; Marshall-Keim, T.; Khomh, F.; Moir, K. The Practice and Future of Release Engineering: A Roundtable with Three Release Engineers. *IEEE Softw.* **2015**, *32*, 42–49. [CrossRef]
22. Barrak, A.; Eghan, E.E.; Adams, B.; Khomh, F. Why Do Builds Fail?—A Conceptual Replication Study. *J. Syst. Softw.* **2021**, *177*, 110939. [CrossRef]
23. Herbsleb, J.D.; Mockus, A. Formulation and Preliminary Test of an Empirical Theory of Coordination in Software Engineering. *SIGSOFT Softw. Eng. Notes* **2003**, *28*, 137–138. [CrossRef]
24. Conway, M.E. How Do Committees Invent? *Datamation* **1968**, *14*, 28–31.
25. Bailey, S.E.; Godbole, S.S.; Knutson, C.D.; Krein, J.L. A Decade of Conway's Law: A Literature Review from 2003–2012. In Proceedings of the 2013 3rd International Workshop on Replication in Empirical Software Engineering Research, Baltimore, MD, USA, 9 October 2013; pp. 1–14.
26. Begel, A.; Khoo, Y.P.; Zimmermann, T. Codebook: Discovering and Exploiting Relationships in Software Repositories. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering—Volume 1*; Association for Computing Machinery: New York, NY, USA, 2010; pp. 125–134.
27. Dingsoeyr, T.; Falessi, D.; Power, K. Agile Development at Scale: The Next Frontier. *IEEE Softw.* **2019**, *36*, 30–38. [CrossRef]
28. Herbsleb, J.D.; Moitra, D. Global Software Development. *IEEE Softw.* **2001**, *18*, 16–20. [CrossRef]
29. Crowston, K.; Wei, K.; Howison, J.; Wiggins, A. Free/Libre Open-Source Software Development: What We Know and What We Do Not Know. *ACM Comput. Surv.* **2008**, *44*, 35. [CrossRef]
30. Palomba, F.; Andrew Tamburri, D.; Arcelli Fontana, F.; Oliveto, R.; Zaidman, A.; Serebrenik, A. Beyond Technical Aspects: How Do Community Smells Influence the Intensity of Code Smells? *IEEE Trans. Softw. Eng.* **2021**, *47*, 108–129. [CrossRef]
31. Joblin, M.; Apel, S.; Hunsen, C.; Mauerer, W. Classifying Developers into Core and Peripheral: An Empirical Study on Count and Network Metrics. In Proceedings of the 2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE), Buenos Aires, Argentina, 20–28 May 2017; pp. 164–174.
32. Martini, A.; Bosch, J.; Chaudron, M. Investigating Architectural Technical Debt Accumulation and Refactoring over Time: A Multiple-Case Study. *Inf. Softw. Technol.* **2015**, *67*, 237–253. [CrossRef]
33. Benlian, A.; Pinski, M.; Adam, M. Team-Enacted Use vs. Developer-Needed Use of Agile Practices: How Perceptual (In-)Congruence and Team Feedback-Seeking Shape Developer Well-Being. *Inf. Syst. Res.* **2025**, *36*, 2253–2277. [CrossRef]
34. Mauerer, W.; Joblin, M.; Tamburri, D.A.; Paradis, C.; Kazman, R.; Apel, S. In Search of Socio-Technical Congruence: A Large-Scale Longitudinal Study. *IEEE Trans. Softw. Eng.* **2022**, *48*, 3159–3184. [CrossRef]
35. Forsgren, N.; Humble, J.; Kim, G. *Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations*; IT Revolution: Portland, OR, USA, 2018.
36. DORA. *Accelerate State of DevOps Report 2023*; Google Cloud: Dallas, TX, USA, 2023.
37. Gousios, G.; Pinzger, M.; van Deursen, A. An Exploratory Study of the Pull-Based Software Development Model. In *Proceedings of the 36th International Conference on Software Engineering*; Association for Computing Machinery: New York, NY, USA, 2014; pp. 345–355.
38. Eibl, G.; Thurnay, L. The Promises and Perils of Open Source Software Release and Usage by Government—Evidence from GitHub and Literature. In Proceedings of the 24th Annual International Conference on Digital Government Research, Gdańsk, Poland, 11–14 July 2023; pp. 180–190.
39. Kalliamvakou, E.; Gousios, G.; Blincoe, K.; Singer, L.; German, D.M.; Damian, D. The Promises and Perils of Mining GitHub. In *Proceedings of the 11th Working Conference on Mining Software Repositories*; Association for Computing Machinery: New York, NY, USA, 2014; pp. 92–101.
40. Coelho, J.; Valente, M.T. Why Modern Open Source Projects Fail. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*; Association for Computing Machinery: New York, NY, USA, 2017; pp. 186–196.
41. Wattanakriengkrai, S.; Chinthanet, B.; Hata, H.; Kula, R.G.; Treude, C.; Guo, J.; Matsumoto, K. GitHub Repositories with Links to Academic Papers: Public Access, Traceability, and Evolution. *J. Syst. Softw.* **2022**, *183*, 111117. [CrossRef]
42. Rastogi, A.; Nagappan, N.; Gousios, G.; van der Hoek, A. Relationship Between Geographical Location and Evaluation of Developer Contributions in GitHub. In *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*; Association for Computing Machinery: New York, NY, USA, 2018; pp. 1–8.

43. Weiss, C.; Premraj, R.; Zimmermann, T.; Zeller, A. How Long Will It Take to Fix This Bug? In Proceedings of the Fourth International Workshop on Mining Software Repositories (MSR'07:ICSE Workshops 2007), Minneapolis, MN, USA, 20–26 May 2007; p. 1.
44. Okhuysen, G.A.; Bechky, B.A. 10 Coordination in Organizations: An Integrative Perspective. *Acad. Manag. Ann.* **2009**, *3*, 463–502. [CrossRef]
45. Argote, L.; Miron-Spektor, E. Organizational Learning: From Experience to Knowledge. *Organ. Sci.* **2011**, *22*, 1123–1137. [CrossRef]
46. Hinds, P.J.; Bailey, D.E. Out of Sight, Out of Sync: Understanding Conflict in Distributed Teams. *Organ. Sci.* **2003**, *14*, 615–632. [CrossRef]
47. Strode, D.; Dingsøyr, T.; Lindsjorn, Y. A Teamwork Effectiveness Model for Agile Software Development. *Empir. Softw. Eng.* **2022**, *27*, 56. [CrossRef]
48. Rigby, P.C.; Bird, C. Convergent Contemporary Software Peer Review Practices. In *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*; Association for Computing Machinery: New York, NY, USA, 2013; pp. 202–212.
49. Mishra, D.; Mishra, A.; Ostrovska, S. Impact of Physical Ambiance on Communication, Collaboration and Coordination in Agile Software Development: An Empirical Evaluation. *Inf. Softw. Technol.* **2012**, *54*, 1067–1078. [CrossRef]
50. Choudhary, S.; Bogart, C.; Rose, C.; Herbsleb, J. *Modeling Productivity in Open Source GitHub Projects: A Dataset and Codebase*; Carnegie Mellon University: Pittsburgh, PA, USA, 2020. [CrossRef]
51. Wooldridge, J.M. *Econometric Analysis of Cross Section and Panel Data, Second Edition*; MIT Press: Cambridge, MA, USA, 2010.
52. Angrist, J.D.; Pischke, J.-S. *Mostly Harmless Econometrics: An Empiricist's Companion*; Princeton University Press: Princeton, NJ, USA, 2009.
53. Cameron, A.C.; Miller, D.L. A Practitioner's Guide to Cluster-Robust Inference. *J. Hum. Resour.* **2015**, *50*, 317–372. [CrossRef]
54. Reagans, R.; Argote, L.; Brooks, D. Individual Experience and Experience Working Together: Predicting Learning Rates from Knowing Who Knows What and Knowing How to Work Together. *Manag. Sci.* **2005**, *51*, 869–881. [CrossRef]
55. Dabbish, L.; Stuart, C.; Tsay, J.; Herbsleb, J. Social Coding in GitHub: Transparency and Collaboration in an Open Software Repository. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*; Association for Computing Machinery: New York, NY, USA, 2012; pp. 1277–1286.
56. MacKinnon, D.P.; Krull, J.L.; Lockwood, C.M. Equivalence of the Mediation, Confounding and Suppression Effect. *Prev. Sci.* **2000**, *1*, 173–181. [CrossRef]
57. Harms, C.; Lakens, D. Making "null Effects" Informative: Statistical Techniques and Inferential Frameworks. *J. Clin. Transl. Res.* **2018**, *3*, 382–393.
58. Parnin, C.; Rugaber, S. Resumption Strategies for Interrupted Programming Tasks. *Softw. Qual. J.* **2011**, *19*, 5–34. [CrossRef]
59. Kuutila, M.; Mäntylä, M.; Farooq, U.; Claes, M. Time Pressure in Software Engineering: A Systematic Review. *Inf. Softw. Technol.* **2020**, *121*, 106257. [CrossRef]
60. Kudaravalli, S.; Faraj, S.; Johnson, S.L. A Configural Approach to Coordinating Expertise in Software Development Teams. *MIS Q.* **2017**, *41*, 43–64. [CrossRef]
61. Luo, C.; Xiang, L.; Hu, J.; Yang, K. Bedrock Models in Communication and Sensing: Advancing Generalization, Transferability, and Performance. *arXiv* **2025**, arXiv:2503.08220. [CrossRef]