

Article

Negotiating Completion in Agentic Software Work: Acceptance Boundaries and Deferred Requirements

Minyang Ling, Ju Zou * and Xintong Liu

School of Journalism and Communication, Nanjing Normal University, Nanjing 210097, China

* Correspondence: njnuzouju@gmail.com

Highlights

Links to systems science

- The study treats software completion as a sociotechnical process connecting review decisions, evaluation artifacts, and continuing work.
- Its process model explains how feedback changes implementation, accepted scope, or the evidence used to judge completion.

What are the main findings and implications?

- Review establishes which requirements bind the current contribution and which continue through linked issues or alternative implementations.
- Test configurations enact acceptance decisions, while later repairs and rollbacks can change the requirements an accepted version addresses.

Abstract

Coding agents increasingly participate in software development, but submitting an implementation does not settle whether work is complete. This qualitative process comparison examines 30 diagnostic pull requests from 22 GitHub repositories within a frame of 1782 contributions from 120 repositories. Manual screening found substantive contestation in 516 frame contributions (29.0%). The analysis draws on public discussions, reviews, commits, tests, and linked artifacts to trace how participants address unmet or unproven requirements. Using acceptance boundary as a sensitizing distinction, the analysis connects conditions stated in review to their enactment in evaluation artifacts and subsequent revision. Participants may retain requirements for repair, delimit a smaller acceptable increment, pursue an alternative implementation, or terminate an approach. Tests enact these decisions through repair, authorized exclusion, and restoration. In squuff, issue closure accompanied partial repair and renewed deferral; a subsequent rollback reversed the accepted changes. Integration, issue closure, and requirement satisfaction therefore represent distinct accomplishments. The process model identifies conditional, overlapping paths connecting local acceptance to continuing work. It explains collaborative completion in agent-associated software development and shows how documented interface constraints and human intervention shape subsequent work.



Academic Editors: Mathias Fonkam
and Narasimha Rao Vajjhala

Received: 9 September 2026

Revised: 10 October 2026

Accepted: 12 October 2026

Published: 16 October 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article

distributed under the terms and

conditions of the [Creative Commons](#)

[Attribution \(CC BY\)](#) license.

Keywords: agentic AI; software collaboration; sociotechnical systems; pull requests; acceptance criteria; articulation work; process research

1. Introduction

Coding agents produce software changes that other participants must assess for integration into a shared system. Contributions include code, tests, and accounts of what has been accomplished, but review raises a further question: what makes the work complete enough to integrate? The answer depends on implementation, available evidence, and which requirements participants assign to the current contribution.

These dependencies make acceptance a sociotechnical problem. Systems engineering emphasizes the relations between technical arrangements and organizational practices [1,2]. Here, contributors and maintainers work with generated implementations, automated checks, and records of unfinished work. An integration decision changes the codebase while establishing what can count as finished within a larger undertaking.

The material contains a revealing contrast. One reviewer instructed an agent to preserve and fix failing timer tests; another collaborator permitted failing dialect tests to be disabled. Both addressed unmet expectations but gave them different roles in acceptance. The dialect trajectory continued through partial restoration and then rollback. Interpreting integration and issue closure therefore requires tracing the requirement's state across versions.

Delegation research addresses ambiguous tasks and distributed coordination [3,4]. Software collaboration research shows that discussion changes requirements and that accepted alternatives can supersede rejected contributions [5,6]. Building on these findings, this study follows how acceptance decisions are enacted through tests and how remaining requirements enter subsequent work.

The study asks: How do participants handle unmet or unproven requirements in contested agent-associated contributions as work proceeds toward acceptance, continuation, or termination? A comparative analysis examines 30 diagnostic pull requests (PRs) from 22 repositories within a sampling frame of 1782 PRs from 120 repositories. The cases provide contrasts in review, author participation, evidence requests, scope decisions, and final dispositions. Explicit links to issues and subsequent contributions allow selected requirements to be followed beyond the original PR.

The analysis uses acceptance boundary to distinguish what must be satisfied or evidenced for the current contribution from what is assigned another disposition. The cases add a process relation to this initial distinction: a stated acceptance condition becomes consequential through changes to an evaluation artifact, and the resulting settlement can be revised as the same requirement enters later work. This connects delegation to the temporal organization of unfinished work: tests enact the current decision, while linked records carry excluded requirements into subsequent contributions. The contribution is an account of collaborative completion observed in agent-associated work. Its relevance to agent-associated work lies in documented interface constraints and changes in who can carry out a revision.

2. Theoretical Background

2.1. Delegation and the Problem of Completion

Delegation provides a starting point for studying agentic information systems. Baird and Maruping conceptualize it through appraisal, distribution, and coordination, including situations in which task requirements are ambiguous [3]. Their framework makes completion an organizational question: producing an output does not establish participants' agreement that the delegated work has been accomplished.

Distributed delegation extends this coordination beyond a single person and artifact. Stelmaszak and colleagues identify collective coordination, adaptive alignment, and intervention in human–algorithm interaction at Uber [4]. Software development likewise

connects submitters, reviewers, assistants, test infrastructure, and maintainers. Their interdependence raises the question of what they accept when they allow a contribution to proceed.

Research on productive delegation examines the allocation of tasks across human and AI capabilities [7], while delegation design makes verifiability explicit [8]. Review shows how these considerations operate when participants dispute the meaning, feasibility, or evidentiary basis of a completion criterion during execution.

A systems perspective locates completion criteria within joint work [9]. Research on machines as teammates and human–AI synergy examines the practical arrangements that support collaboration [10,11]. Contribution review connects the proposed change, its evaluation, and records of unresolved work, linking local acceptance to what remains necessary in the shared project.

2.2. *Evaluating Contributions and Transforming Work Objects*

Negotiation over software contributions predates contemporary coding agents [12]. Tsay and colleagues found that GitHub discussions address the appropriateness of the problem, the implementation of the solution, and the evolution of project requirements [5]. Their study also shows that an alternative implementation can fulfill a submitter’s technical goal even when the original contribution is rejected. Steinmacher and colleagues similarly identify superseded contributions and disagreements about project direction among the reasons that contributions remain unaccepted [6]. These findings locate requirement negotiation and contribution replacement within the established organization of cooperative software work.

Studies of agent-associated contributions distinguish replacement, other rejection reasons, and cases without explicit closing explanations [13,14]. PatchTrack also documents ChatGPT-assisted PRs superseded by refined implementations [15]. Following a disputed requirement across such connections can establish what participants carried forward, revised, or set aside.

Recent studies identify further variation in the processes leading to agentic PR outcomes. Ehsani et al. identify failure patterns involving review participation, duplicate work, and implementation or integration problems [16]. Hindi et al. examine failure modes and rejection patterns in agent-generated contributions [17]. Strauss and Schulze relate acceptance to combinations of contextual factors, including PR size and integrator response time [18]. Peralta et al. distinguish substantive failure from workflow-related rejection and cases with no observable reason for rejection, while documenting human involvement in merged work [19]. These studies make clear why disposition alone cannot establish task accomplishment. A testing-focused comparison by Milanese et al. finds broadly comparable frequencies of test inclusion in human–agent and human PRs, alongside differences in the extent and type of test changes [20]. The present analysis follows a different empirical object within this established problem: how a particular requirement becomes a condition of acceptance, how code or tests enact that condition, and what happens to the requirement after integration or replacement.

Articulation work explains how activities within a common undertaking are connected [21,22]. Implementation, assessment, and later correction require coordination across participants and time [23,24]. Follow-up issues sustain that connection by recording unresolved requirements alongside accepted changes [25].

Coordination can also transform the objects being assessed. Translational mobilization theory explains their collective construction and stabilization [26], while knowledge and boundary-object research emphasizes the interests and dependencies accommodated in

shared artifacts [27,28]. An accepted increment and a separate record of residual requirements form a pair of connected objects whose histories may subsequently diverge.

Technical-debt and test-maintenance studies identify concrete forms of residual work. Code review can introduce self-admitted technical-debt comments, including at reviewers' request [29]. Disabled tests may return to use and be disabled again, with histories diverging from issue status [30]. These findings support examining how requirements become binding or deferred and how evaluation artifacts enact and later revise those decisions.

The term *acceptance boundary* is used here to denote the distinction participants establish between requirements that must be satisfied or evidenced for the current integration decision and requirements assigned another disposition. A disposition may involve explicit deferral, a different implementation, or a decision to abandon the proposed approach. This is a sensitizing distinction for comparing events. Its empirical content comes from participants' requests and decisions, code and test changes, and links between work artifacts. Identifying an explicitly negotiated boundary therefore requires relating the decision to the requests and changes through which participants established it. Scope negotiation identifies what belongs in the work, prioritization orders its urgency, technical debt concerns acknowledged residual work and its costs, and incremental development organizes delivery over successive changes. The acceptance-boundary distinction directs attention to the point at which those arrangements become obligations for a specific integration decision. Permission to omit a requirement is established through explicit agreement and its enactment in the test configuration. The empirical question is how this relation is enacted and subsequently revised.

The distinction depends on what participants make a condition of acceptance. An original task requirement or a review-added test can become binding for the current contribution. A suggestion recorded for possible later development has a different status until participants make it part of that obligation. Similarly, withdrawing support for an implementation strategy ends that proposed way of addressing a problem; the underlying requirement may remain open. These differences make it necessary to follow both the content of a request and the decision about where it belongs.

2.3. Evidence and the Capacity to Act on Feedback

Assessing a contribution depends on practices that establish which claims the evidence warrants [31]. AI-related knowledge-work research examines the interrogation of outputs and changes in knowledge production and recognition [32–34]. In review, a test can pass without distinguishing the intended behavior from its absence, making the adequacy of the test itself a question for investigation.

Dhanorkar and colleagues identify planning, monitoring, and review in developers' oversight of software agents, including the use of passing tests as a heuristic for correctness [35]. Following disputed tests through revision extends this account to how participants change the basis of judgment and organize the further work that oversight exposes.

Acting on such judgments depends on how the work of producing and assessing outputs is organized. Research on automation and augmentation identifies continuing dependencies on human activity [36], including the evaluation and corrective work examined in the ironies of automation and generative AI [37,38]. For contested contributions, the consequential question is what this activity changes: an implementation, the evidence used to assess it, or the scope accepted in the current increment. Each has a different relation to the requirements remaining after review.

A request becomes consequential when someone can carry out the relevant action. Coactive Design emphasizes the interdependence that must be supported in joint activity [39]. In a repository, responding to a comment, editing a branch, stopping an integration,

and approving a contribution are distinct actions. Their distribution can shape how an acceptance boundary is implemented. This consideration complements the analysis of requirements and evidence: a plausible settlement must also be enacted through the available people, accounts, and tools [40].

3. Materials and Methods

3.1. Research Setting and Comparative Design

The study compares publicly documented review trajectories on GitHub. PRs combine proposed changes with discussion, reviews, revisions, and approval or closure. Linked issues and subsequent PRs extend some trajectories, allowing statements about completion to be related to observable changes in the work.

The sampling frame comprised 1782 agent-associated PRs from 120 repositories, drawn from the AIDev source data [41]. The source pool contained 33,596 PRs from 2807 repositories with at least 100 stars in the source records. The selected frame covers PRs created between 24 December 2024 and 30 July 2025.

The frame combined a 100-repository core with 20 repositories selected for information-rich interaction. Repositories were selected to vary in language, star band (100–999, 1000–9999, and 10,000 or more), and observable non-author participation. Participation was used to vary the frame, which included 36 repositories with no returned non-author accounts classified as User in the source interaction tables. Within repositories, selection sought variation in outcome, agent association, interaction, and time, with at most 20 PRs per repository. From this frame, iterative purposive reading and comparison produced the final 30-case set. Selection favored traceable requirements and consequential responses, then included related tasks with different dispositions and cases that challenged or qualified an emerging interpretation. Both contestation and known outcomes entered selection. Supplementary Material S2 specifies the selection rules and comparative uses.

The final diagnostic set comprised 30 PRs from 22 repositories. Its suitability for addressing this process question rests on the available contrasts in retained requirements, enacted deferral, linked replacement, direct human intervention, and limited public visibility. These contrasts allow consequential transitions and their conditions to be examined across settings. Table 1 identifies the empirical levels used in the analysis.

Table 1. Empirical design and analytical use.

Empirical Level	Material	Analytical Use
Sampling frame	1782 PRs from 120 source repositories	Locate varied review settings and contrasting outcomes
Diagnostic cases	30 PRs from 22 repositories; 15 merged and 15 closed without merging	Compare contested requirements, responses, and dispositions
Process evidence	PR descriptions, issue discussions, reviews, inline comments, commits, selected diffs, checks, and linked work artifacts	Reconstruct observable transitions and distinguish statements from actions
Evidence index	152 selected event anchors with original identifiers, timestamps, and source links	Connect interpretations to identifiable public records

We manually screened all 1782 PRs to characterize substantive contestation in the sampling frame. Of all 1782 PRs, 516 showed substantive contestation (29.0%), 1170 showed no substantive contestation in the available records (65.7%), and 96 were indeterminate (5.4%). Indeterminate cases remain in the denominator. The screening characterizes the frame as a whole; the diagnostic cases were selected through the purposive comparisons

described above. An absence of substantive objection differs from an absence of interaction and from an absence of an explicit reason for rejection.

Cases were selected for differences that could challenge an explanation based on continued author participation or successful technical repair alone. They include sustained responses followed by closure, intervention by another participant followed by integration, disputes over the adequacy of tests, scope reductions, and explicit replacement through another PR. Contrasts within the same repository in *sqruff*, *shopware*, and *openai/codex* provide comparisons within shared project settings. Continued responses followed by closure challenge an account in which responsiveness is sufficient for integration. Minimal visible negotiation tests the reach of a model of contested acceptance. Cases with unexplained closure provide evidence about preceding interactions while leaving the terminal decision unresolved. Section 4.5 maps all 30 cases to the process interpretation; Table A1 identifies their source agent labels and recorded outcomes.

Agent association follows the source classification. A PR opened by a human account can include agent-assisted work, and an account associated with an agent can participate alongside people who alter the contribution. Consequently, the analysis concerns trajectories of agent-associated work. The unit of interpretation is a disputed requirement and its disposition within a PR or an explicitly connected sequence of artifacts. A subsequent PR enters a sequence when a participant identifies it as a replacement or continuation, or when a closing relation identifies the task it addresses. A PR can contain several requirements with different dispositions; each is followed separately. Connected PRs concerning the same requirement are interpreted as segments of a shared trajectory. Source agent labels describe the empirical setting; consequential changes are attributed through event and commit records.

3.2. Event Reconstruction and Evidentiary Distinctions

The event records retain repository names, PR and issue numbers, native comment and review identifiers, commit identifiers, account types, recorded author associations, timestamps, and source URLs. Events were ordered chronologically by their recorded timestamps, preserving the distinction between an event occurring before a final decision and a later retrospective statement. The observation cutoff is 3 September 2026. Repository associations and account types describe the public record; actions such as requesting a revision or merging a PR provide the basis for describing participants' roles in a particular episode.

Each case account relates the proposed task to subsequent challenges, requests for evidence or modification, visible responses, code changes, and recorded disposition. For frame screening, substantive contestation means a specific objection to the proposed behavior, implementation, evidentiary basis, or integration scope of the current contribution. Coders read ordinary comments, formal reviews, and inline comments in context. A request to restore a skipped test qualifies even without a `CHANGES_REQUESTED` label. Thanks, unqualified approval, purely presentational edits, automated status notices, and optional future suggestions do not by themselves qualify; neither comment counts nor CI failures substitute for semantic judgment. A record was indeterminate when the available material could not support a judgment. Mirrored representations of a native event were consolidated using its original identifier.

Several distinctions govern the interpretation. An author's statement that tests pass is a completion claim; an available check result is an observation tied to its recorded commit and time. A patch demonstrates a code change, while a participant's report of running an experiment establishes what that participant reported. A merged PR records integration into the repository. A closed issue records an administrative disposition whose substantive meaning must be examined through associated changes. An explicit replacement links

the original contribution to the subsequent pursuit of its task. These distinctions allow apparently similar outcomes to be compared at the level of the requirement being discussed. Agent association, the account submitting a commit, and the means used to produce its contents are distinct. The accounts below name roles and attribute consequential actions where the record supports them. A maintainer commit after an integration was disabled is treated as direct human intervention.

Interpretation connects statements to enactment: squiff's deferral instructions are read alongside test exclusions, and sequin's reported diagnostic experiment with the final test code. Where an explicit closing explanation is absent, the cause of closure remains unspecified.

A requirement's recorded state is tied to a dated revision. The reconstruction records test configuration and observed execution separately, and follows linked reversals before carrying an earlier state forward. Reverts, reopenings, and replacement links enter the same trajectory when they concern the change or requirement under interpretation. The 152 event anchors in S1 index the evidence used in the case interpretations. Connected patches and later disposition records establish the requirement's state at the next observed revision. A trajectory extends through explicit task links or identifiable changes to the same requirement and ends at the last such event dated on or before the observation cutoff. The frame includes 1658 PRs with current records and 124 represented by source snapshots. Timeline coverage is incomplete for 28 of the 30 diagnostic cases. Comments and reviews were assessed separately for coverage, and interpretation rests on documented events; a missing timeline event does not establish that an action did not occur. Supplementary Material S2.3 provides the coverage assessment, including its relation to the zero-comment case in Table S21.

3.3. Analytical Strategy

The analysis follows the logic of process theorizing: reconstruct situated sequences, compare consequential transitions, and develop an explanation that preserves their temporal relations [42]. Initial description focused on what participants claimed, questioned, requested, and changed. Acceptance boundary supplied a sensitizing distinction; the process categories were developed by comparing the treatment and enactment of requirements across these sequences. Retained obligations accompanied repair, explicit scope decisions moved work beyond the current increment, and reconsideration redirected or ended an implementation. Comparing later revisions added the relation among repair, renewed deferral, and reversal.

The comparison distinguishes the proposed behavior, the means of implementing it, and the evidence requested to assess it. Each request is located in the original task description or subsequent review, together with any statement making it a condition of the current contribution. Requirements binding on the current contribution were identified from conditions stated before or at the acceptance decision and from the actions that implemented them. Authorization to defer was identified in a participant's explicit instruction or agreement, read together with that participant's role and the ensuing response or revision. Suggestions for later enhancement were classified according to the obligation participants actually established. Requirements are linked across artifacts through their stated behavior, test or rule identifiers, and explicit task connections. In squiff, for example, the AL07 identifier and its CREATE TABLE test connect successive explanations of the same exclusion. Section 4.5 presents the analytical uses of the 30 PRs; Supplementary Material S2 provides evidence entry points and coding rules, and S3 records the relevant requirement versions. The PR is a case-selection unit; the requirement trajectory is the unit through which its treatment is interpreted. Linked PRs are interpreted as parts of a connected trajectory. For consistent application, path A requires positive evidence of

a retained requirement and an associated repair to implementation or evidence. Path B requires explicit authorization to narrow or defer the current obligation and evidence of its enactment. Path C requires an explicit replacement, redirection, or termination of the approach. A request with only a partly documented response is coded as a partial process. A passing-test claim remains reported evidence. Paths can overlap within a case; coding records positive support for each path. For example, git-link documents authorization to separate a simulation, but the selected record does not independently establish the removal patch; its deferral evidence is therefore partial. Sqruff connects permission directly to ignored-test markers and a remaining-work record.

Competing accounts were considered at the point where they could explain the same transition: technical repair, ordinary incremental development, changes in project priorities, and differences in contributors' ability to act. The interpretation relates each consequential decision to the preceding request, its enactment, and the subsequent disposition recorded in S1–S3. The authors reconstructed the cases and examined the requirement-level interpretations against the GitHub records. Authors and research assistants conducted the manual screening and independent case review. One coder screened all 1782 PRs; a second independently screened a simple random sample of 535 (approximately 30%). The coders' initial independent judgments agreed on 500 PRs (93.46%; unweighted Cohen's $\kappa = 0.8615$), with 35 disagreements subsequently resolved. All 30 diagnostic cases received two independent evidence reviews using the requirement and pathway rules. Six cases had at least one disputed interpretation; a third reviewer adjudicated these, changing at least one interpretation relative to the first review in four cases. This case review is separate from the 535-PR screening agreement. S2.3 reports the rules and the initial screening cross-tabulation.

4. Results

4.1. Preserving or Redrawing the Current Requirement

In primer/react #6416, Copilot was migrating the SelectPanel test suite from Jest to Vitest. The work involved carrying existing tests into a different browser testing environment, including tests of timed screen-reader announcements. On 30 July 2025, the agent responded to a request to replace visibility assertions with assertions of presence in the document. Its completion report stated that 42 tests passed, explicitly excluding five skipped timer tests. The report made the migration appear successful within the set of tests still being run, while identifying behavior that the migrated suite no longer checked.

The reviewer's intervention focused on the skipped tests. At 06:56 UTC, the instruction was to "dont skip the timer tests" and to make the changes needed to fix them. Fourteen minutes later, the agent reported replacing fake timers with real timers and waiting utilities, identified commit 84662b4, and claimed that all 47 tests now passed. The response matters as an account of what the agent understood the review to require: the five tests had to be reinstated to meet the migration's completion criteria. Within the migration, the reviewer accepted an alteration to visibility assertions while insisting that the timer checks remain. A successful run of the reduced suite had not satisfied the latter requirement. The distinction concerned which behavior still had to be checked as the testing method changed.

quarylabs/sqruff #1664 introduced T-SQL, the Microsoft SQL Server dialect, into a SQL linter and formatter. Extracted from a broader implementation, it separated dialect grammar from rule changes. That distinction mattered: support for SQL syntax could advance while linting or formatting behavior still failed its tests.

On 21 June, the collaborator authorized a way to proceed: the author could "comment out the failing test" and remove SQL files that remained unparsable. The following evening, the author explained that two issues had been created for problems that could not be fixed,

that other changes had been made, and that unparsable test files had been removed. The author linked this action to the expectation that continuous integration (CI) would pass. On 23 June, the author reported disabling further tests and, before the merge, connected the outstanding limitations to issue #1685. These exchanges established a basis on which a narrower increment could be accepted despite acknowledged unsupported behavior.

The decision took material form in commit 4157492. Ignored-test entries described problems such as unsupported CREATE TABLE syntax and function-name capitalization, while a known-issues document identified affected constructs and rules. The entries performed two connected operations. They marked particular tests to be skipped, consistent with the author's stated expectation that CI would pass, and retained descriptions of the behavior that still required work. The author's description of T-SQL support as experimental gave those exclusions a place within the increment being offered. The collaborator merged the PR on 23 June, after the deferrals had been stated and implemented.

The task gave test exclusion its significance. Primer's migration required existing checks to be carried forward; squuff's experimental extension allowed specified behavior to remain outside the current increment. A similar testing operation therefore elicited repair in one setting and enacted authorized deferral in the other. The acceptance boundary took effect through the relation between the project decision and test configuration.

sshaw/git-link (<https://github.com/sshaw/git-link/pull/144>) (accessed on 15 July 2026) #144 shows how that boundary could change within a single review. The contribution added tests for functions that generate links to repository providers. During review on 12 July, the owner requested an interactive test that would open a buffer, select a line, and invoke git-link as a user would. Subsequent comments asked about avoiding mocks and checking the entire generated URL. These requests expanded the evidence expected of the contribution beyond testing individual provider functions. They sought a demonstration of the user-facing operation.

When the simulation continued to fail, the owner asked to remove it so the other tests could be integrated and the simulation pursued separately. Six minutes later, the owner prohibited changes to git-link.el, keeping the contribution confined to tests. After merging at 23:53 UTC, the owner repeated the request for a separate simulation PR. These instructions specified both an omission and a constraint on expanding the contribution. They establish scope authorization, but the selected evidence does not independently verify the removal patch; the deferral sequence therefore remains partial.

The simulation remained desirable after merge. Separating it changed its relation to current acceptance without declaring it unnecessary, preserving a request for further work.

Two further repairs show how project knowledge entered the revision. In airbytehq/airbyte #52636, a reviewer traced a documentation failure to the handling of headingIndex after the Docusaurus change; subsequent parsing revisions and confirmation preceded merge. In microsoft/onnxscript #2455, review addressed an inputs field, MatchContext naming, and lint requirements. Both an agent lint commit and a human pylint change are visible. These sequences connect specific objections to modifications while showing that repair involved contributions from more than one actor.

4.2. Deferred Work Can Be Repaired and Repartitioned

Two requirement trajectories run through the work on the same dialect. The test-exclusion and rollback trajectory follows which checks the accepted increment exercised. The CV07 replacement trajectory follows a separate rule requirement into an alternative implementation. Each trajectory follows a distinct requirement within the shared issue context.

The linked squuff records allow this further work to be traced. Issue #1685 gathered the limitations associated with the disabled T-SQL tests. Three days after #1664 merged, PR #1698 was integrated, and its closing commit d59b454 closed the issue. The associated changes addressed operators, CREATE statements, and other constructs implicated in the earlier failures. The patch therefore provides a substantive connection between the recorded deferral and later implementation work.

At the level of individual tests, that connection had different outcomes. In this revision, explicit ignore markers were removed from the CP01 function-capitalization test and the CV01 comparison-operator tests. For the AL07 rule, however, the marker remained. Its explanation changed from incomplete support for T-SQL CREATE TABLE syntax to the need for the AL07 rule itself to handle CREATE TABLE statements. An LT02 test likewise remained ignored, but its explanation shifted from unsupported CREATE FUNCTION syntax to an indentation problem requiring changes in the reflow system. Reflow concerns how code is laid out, so improving the parser did not by itself settle the formatting requirement.

At integration on 26 June, issue closure accompanied partial restoration and revised explanations for continuing exclusions. The patch separated problems previously grouped as incomplete language support, identifying more specific locations for later repair. Acceptance incorporated that progress while retaining obligations beyond the increment.

That configuration changed again on 28 June. The collaborator merged PR #1718, explicitly reverting the #1698 commit d59b454. The revert reinstated the ignored marker for the CP01 function-capitalization test and the two CV01 comparison-operator tests. It also returned the AL07 and LT02 explanations to the earlier parsing limitations. The changes observed on 26 June therefore describe an accepted revision whose relevant test configuration was subsequently reversed.

PR #1719 proposed undoing this rollback but was not merged. In July 2025, discussion on that PR turned toward prioritizing the dialect's parsing problems; the PR was subsequently closed in July 2026. The trajectory records this as a proposed reinstatement followed by a change in work priority.

A distinct requirement in the same T-SQL undertaking continued through PR #1706: removing unnecessary outer parentheses under rule CV07. Its description explicitly connected this problem to #1685 and identified it as work not fixed in #1698. On 27 June, the author described an intention to address individual linting issues and asked collaborators for implementation suggestions, explaining that available time was limited. The request clarified the practical circumstances of the next attempt: a contributor using Claude was trying to advance a collection of related requirements and seeking project-specific direction on how to do so.

On 28 June at 22:01 UTC, the collaborator stated that the unnecessary wrapping had been fixed and immediately linked PR #1721. That alternative merged at 22:48. The author endorsed it the next morning, and #1706 closed seconds after that response. The relationship between the two PRs is established by the proposal of the alternative, its integration, and the original author's endorsement. Work on the requirement continued through another implementation while the original PR was closed. Table 2 locates this replacement within the broader dialect trajectory.

The two trajectories show why following an undertaking requires attention to individual requirements and revisions. Selected tests were restored and then excluded again, while work on CV07 proceeded through an alternative implementation. Their shared task thus encompassed divergent requirement histories.

Table 2. Two requirement trajectories in the squuff undertaking.

Requirement Trajectory	Version and Date (UTC)	Recorded Change	Interpretation
Test exclusion and rollback	21–23 June 2025; #1664	Deferral authorized; tests disabled; #1685 records limitations	Current acceptance permits specified unsupported behavior
Test exclusion and rollback	26 June 2025; #1698, d59b454	CP01/CV01 restored; AL07/LT02 still ignored with revised explanations; issue closed	Partial repair and renewed deferral coexist
Test exclusion and rollback	28 June 2025; #1718, b5466a3	Revert restores CP01/CV01 exclusions and earlier AL07/LT02 explanations	An accepted configuration is reversed
Test exclusion and rollback	13–14 July 2025 discussion; 11 July 2026 closure; #1719	Reinstatement proposed; discussion favors parsing; proposal remains unmerged	Proposed continuation does not establish restoration
CV07 replacement	27–29 June 2025; #1706 → #1721	Separate wrapping requirement; alternative linked and merged on 28 June; original endorsed and closed on 29 June	Task continues through an accepted alternative

4.3. Repairing What Counts as Evidence

In sequinstream/sequin #1795, the contributor connected Sequin’s data pipeline to an S2 destination. Its description listed a client, a connection test, pipeline integration, and accompanying tests, while reporting that tests had not been run because of the execution environment. On 11 June 2025, the contributor reported pushing changes to request stubs, which simulate external requests during testing, and the associated tests, including work to ensure that assertions were executed. The review was therefore already concerned with how to demonstrate that a record sent through the pipeline reached the expected request-handling code.

The next morning, the contributor reported a diagnostic experiment: even after the line where the request should occur was commented out, “the test still passed”. Further investigation pointed to interference between `Req.Test.verify_on_exit!` and `Hammock.verify_on_exit!`, both used to verify expected calls when tests finish. According to the report, disabling the latter caused the test to fail when the request was absent. The contributor traced the difficulty to the interaction between testing components.

The reported experiment undermined the passing result as evidence that the request occurred. Before claiming completion, participants needed an observation that distinguished the target action from its absence independently of the interfering exit checks.

The final test code contains that more direct connection. Inside the expected-request handler, it checks the request and sends a message to the test process; after starting the pipeline and sending a test event, the test asserts receipt of the request message as well as the pipeline acknowledgment. The test thus checks separately that the pipeline acknowledges the event and that the request occurs. The additional assertion requires the expected handler to have been reached, which addresses the specific evidentiary gap described in the review. The patch establishes the form of this check; the account of the historical experiment comes from the contributor who reported it.

The contributor also considered the verification interference “out of scope for this PR” and linked issue #1799 for repair; additional compression support went to #1800. At 08:29 UTC, the contributor retained a batch size of ten and considered the PR ready unless another problem emerged. It merged on 16 June. Acceptance therefore combined the sink’s revised test with separately recorded infrastructure and compression work.

Repairing local evidence made the bounded implementation assessable while preserving a record of the wider infrastructure problem. Interpreting the later closure of those issues requires examining their own resolution records.

In dotCMS/core #32657, the relevant gap emerged through direct use of an interface. The contribution added Swagger/OpenAPI annotations for content-management endpoints. During review on 22 December 2025, a reviewer reported that importing a category-export file produced an HTTP 200 response but created no new categories; the system log showed a null-pointer exception. The apparently successful response and the absent target state were observed within the same operation. Reviewing the annotation therefore required examining what the documented and returned response actually indicated about user-visible behavior.

Further discussion distinguished binary transport from logical API content: removing octet-stream support produced a 406 response, and the author clarified its relation to the annotation. These exchanges moved evaluation from a surface success signal to the behavior it should represent. The PR merged in March 2026, but the reviewed material does not establish a complete repair chain for the failed import.

Other cases specify the evidentiary demand differently. In openai/codex #1547, a reviewer requested an assertion on the whole payload; microsoft/typescript-go #1387 distinguished exercising a code path from demonstrating the intended diagnostic behavior. In AliAkhtari78/SpotifyScraper #21, commit 202e2c5 changed the extractor to the embed endpoint alongside the author's report of testing and restricted access. The PR subsequently merged. The patch establishes a revision, while evidence of its execution outcome remains limited to the available report.

Some interactions concerned how participants could carry out verification. In microsoft/finops-toolkit #1621, a reviewer unfamiliar with running the PowerShell tests received commands and a passing-test report before merge; this exchange was coded as assistance with verification. In openai/codex #1601, approval accompanied a request to address remaining comments, followed by further activity and merge. These sequences show how execution guidance and conditional approval organized continued work.

The evidentiary demand also varied across testing tasks. In dotnet/fsharp #18583, reviewers asked for coverage of List and type members, appropriate placement in the summary, and XML-validation tests. Revisions followed, but the final closure was unexplained. In specmatic/specmatic #1951, a reviewer specified that one row should generate both a number pattern and a string pattern containing "hello," and requested removal of an unnecessary test. The maintainer directly revised tests and implementation. The record establishes the repair, with the cause of subsequent closure remaining unspecified.

4.4. Changing the Implementation or Ending the Approach

The target of review can shift from whether an implementation works to whether its approach remains worth pursuing. shopware/shopware #8269 addressed affiliate and campaign codes that were not stored consistently in production mode. The Devin-associated contribution proposed storing URL parameters in cookies and using them in checkout and registration. This promised persistence for the tracking codes, but it also changed their relationship to the existing session-based handling.

On 4 April 2025, a project member requested unit tests. On 14 April, another asked how the change would affect session storage in the next version and requested feature-flag separation. On 23 April, an outside participant noted that cookie storage required integration with the consent mechanism and proposed sessions as a fallback. The project member questioned that fallback because session handling required backend execution and incurred cache misses.

These exchanges brought together concerns that additional unit tests alone could not settle. Persistent tracking, consent handling, backward compatibility, and the cost of session-based execution implicated different parts of the proposed solution. The suggested fallback addressed one concern while reintroducing a cost that a project member considered undesirable. On 12 May, a project member acknowledged the cookie concern and stated, “I would not follow the approach here anymore”. The PR closed at that time. The stated reason connects the endpoint to a changed assessment of the approach after its dependencies had been discussed. The closing statement withdrew support for cookie-based implementation; the exchange contained no separate decision to abandon the underlying persistence problem.

shopware/shopware #8383 concerned missing search-engine-friendly URLs for landing pages in footer navigation. The maintainer requested tests with those URLs configured: existing cases exercised code without reproducing the condition central to the problem. When the proposed test organization became difficult to follow, the maintainer requested separate methods for each link type, checking results before and after URL configuration.

On 14 April, the maintainer reported disabling the Devin integration and requested reversal of recent commits. The maintainer then addressed review comments and removed an unneeded subscriber on 15 April, made CI fixes on 16 April, and merged the PR that day. The contribution continued through direct human repair under the retained behavioral requirement.

The shopware comparison distinguishes the object of intervention. In #8269, dependencies undermined the proposed cookie strategy; in #8383, the requirement survived while a maintainer changed how revisions were produced. Intervention could therefore terminate an approach or supply the means of completing it.

Arize-ai/phoenix #8660 offers a further instance of direct intervention. Review identified a race condition that UPSERT alone could not resolve and supplied an exception-handling change. A maintainer then committed cleanup and merged the contribution. The observable repair joins an agent-associated proposal to human implementation work.

In primer, the reviewer requested a new PR based on the latest main branch on 1 August. Copilot reported that its interface could not close and open PRs and described the required human actions. Immediately after closing #6416, the reviewer explained that complex merge conflicts required a fresh start. Task issue #6381 subsequently closed through merged #6433, linking the migration to a successor with both agent commits and direct maintainer timer-test repair. The recorded task continued through mixed contributions after the original PR ended.

These explicit relations support a different interpretation from cases whose endpoint remains unexplained. Azure/bicep-registry-modules #5692 names merged #5695 as its replacement. By contrast, the related openai/codex testing PRs include merged #1541 and #1542 and closed #1547 and #1545, despite overlapping authorship and reviewing relationships; the record does not supply an explicit cause for each closure. Continued responses and favorable reviews thus occurred alongside different dispositions. Near/nearcore #13318 supplies a complementary contrast: limited written participation by the author coexisted with continued bot commits and eventual integration.

In dotnet/runtime #115762, build requirements and branch-write permissions constrained the agent’s responses; in microsoft/typespec #7619, review specified the compiler, Generate.ps1, and the emitter build while restricting unrelated changes. Revisions and reported execution limits remained visible in both cases, but neither record supplied an explicit final reason for closure. In dotnet/runtime #115826, race concerns persisted after a reported fix; a policy service later closed the inactive draft. In liam-hq/liam #1380, requests to remove unnecessary null handling were followed by continuing agent commits

and an unexplained closure. Administrative disposition and substantive resolution remain separate in these sequences.

Changes to the project provide an alternative explanation in oven-sh/bun #20298 and #20736: June 2026 closing messages cite a Rust rewrite and altered code organization, making the contributions obsolete. The recorded endpoint followed changes to the project's implementation environment. Conversely, mitsuhiro/insta #750 merged with no discussion returned by its comment and review connections. Its public record documents integration with no observed review exchange.

Table 3 brings the full case set into the comparison. Direct evidence supports A in nine cases, B in two, and C in six; three cases support two paths, yielding 14 cases with at least one direct path. Fifteen other cases document partial processes and one provides a visibility boundary. These overlapping counts describe evidence within the selected PRs. Among the 15 unmerged closures, seven have an explicit recorded reason and eight do not. The recorded reasons encompass substantive decisions, administrative inactivity in the runtime draft, and project change in bun.

Table 3. Requirements, observable paths, actions, and evidentiary limits across all 30 diagnostic PRs.

Case	Requirement or Issue	Path Evidence	Action, Disposition, and Limit
primer/react #6416	Timer checks and migration	A; C	Reviewer retains tests; agent reports repair; mixed successor commits; Closed; task continues in #6433; original pass claim reported
Azure/bicep-registry-modules #5692	Module implementation	A (partial); C	Maintainer names #5695 as replacement; Closed; named successor merged
shopware/shopware #8383	SEO URL behavior and tests	A	Maintainer disables integration, revises tests/code, merges; Merged; direct human repair
sshaw/git-link #144	Interactive simulation	A (partial); B (partial)	Owner authorizes separate work; removal patch not independently verified; Merged; simulation completion unestablished
AliAkhtari78/SpotifyScraper #21	Extraction and verification	A	Agent changes endpoint; reports tests and access limits; Merged; execution remains reported
dotnet/runtime #115762	Build and permissions	A (partial)	Reviewers request build changes; agent revises under permission limits; Closed; reason unknown
dotnet/runtime #115826	Concurrency race	A (partial)	Agent reports fix; reviewer still raises races; policy service closes draft; Closed; inactivity policy, substantive outcome unsettled
dotnet/fsharp #18583	Documentation and XML checks	A (partial)	Reviewers request coverage and tests; agent revises; Closed; reason unknown
near/nearcore #13318	Validation strength	A (partial)	Reviewers reject weakening; bot commits continue; Merged; full repair chain unestablished
Arize-ai/phoenix #8660	Transaction race	A	Reviewer supplies exception handling; maintainer commits cleanup; Merged; direct human intervention
microsoft/finops-toolkit #1621	PowerShell verification	A (partial)	Reviewer seeks execution help; agent gives commands and pass report; Merged; no substantive objection established

Table 3. Cont.

Case	Requirement or Issue	Path Evidence	Action, Disposition, and Limit
microsoft/typespec #7619	Compiler generation and build	A (partial)	Reviewer specifies commands and scope; agent reports constraints; Closed; reason unknown
microsoft/typescript-go #1387	Diagnostic evidence	A (partial)	Reviewers distinguish coverage from correct diagnostics; agent revises; Closed; reason unknown
specmatic/specmatic #1951	anyOf test patterns	A	Reviewer specifies expected patterns; maintainer changes tests/code; Closed; reason unknown despite repair
openai/codex #1601	Conditions attached to approval	A (partial)	Reviewer asks for remaining comments to be addressed; activity continues; Merged; full request-to-change chain unestablished
microsoft/onnxscript #2455	Fields, naming, and lint	A	Reviewer specifies changes; agent and human lint commits; Merged; mixed repair contributions
airbytehq/airbyte #52636	Documentation parsing	A	Reviewer localizes headingIndex problem; agent revises; fix confirmed; Merged; linked repair visible
dotCMS/core #32657	API annotation and import behavior	A (partial)	Reviewer reports HTTP 200 without import; transport discussed; Merged; complete import repair unestablished
quarylabs/sqruff #1706	CV07 wrapping	A (partial); C	Contributor proposes; collaborator links alternative; author endorses; Closed; #1721 merged as alternative
quarylabs/sqruff #1664	T-SQL tests and scope	A; B	Collaborator authorizes exclusion; author changes tests; linked repair/revert; Merged; exclusions partly restored, then rollback
mitsuhiko/insta #750	TTY check	Visibility boundary	Submitting account contributes and merges; no review discussion returned; Merged; visibility boundary
openai/codex #1547	Full-payload assertions	A (partial)	Reviewer requests stronger assertions; author responds and revises; Closed; reason unknown
openai/codex #1541	Assertions and equality support	A (partial)	Reviewer suggests assertions; author explains implementation constraint; Merged; partial requirement evidence
openai/codex #1545	Output assertions and docstrings	A (partial)	Reviewer specifies tests; author continues revisions; Closed; reason unknown
openai/codex #1542	Helpers and test termination	A (partial)	Reviewer raises questions; author submits review revision; Merged; partial request-to-change correspondence
sequinstream/sequin #1795	Discriminating request test	A; B	Contributor diagnoses and changes test; broader issues separated; Merged; local repair and recorded deferral
oven-sh/bun #20298	Redundant checks and code obsolescence	A (partial); C	Reviewer challenges fix; later bot cites Rust rewrite; Closed; explicit project-change explanation

Table 3. Cont.

Case	Requirement or Issue	Path Evidence	Action, Disposition, and Limit
oven-sh/bun #20736	Obsolete code tree	C	Human/agent commits and CI notices; bot cites Rust rewrite; Closed; explicit project-change explanation
liam-hq/liam #1380	Null handling and implementation	A (partial)	Reviewer requests changes; agent continues commits; Closed; reason unknown
shopware/shopware #8269	Cookie-based persistence	A (partial); C	Participants question consent and cost; project member ends approach; Closed; approach abandoned, underlying problem not settled

A: Retain a requirement and repair implementation or evidence. B: Delimit the current increment through enacted, authorized deferral. C: Redirect, replace, or terminate the approach. “Partial” marks an observed segment without the complete evidentiary connection required for direct path support. Unlisted paths lack positive evidence in the reviewed record. S1 supplies source anchors and Table S32 links each case to its evidence entry points.

4.5. A Process Model of Contested Acceptance

Figure 1 relates the treatment of disputed requirements to the contribution’s disposition and subsequent work. The paths can combine: sequin repaired local evidence while separating an infrastructure problem, and shopware paired revision with a change of actor. The model follows sequences of consequential actions within an undertaking.

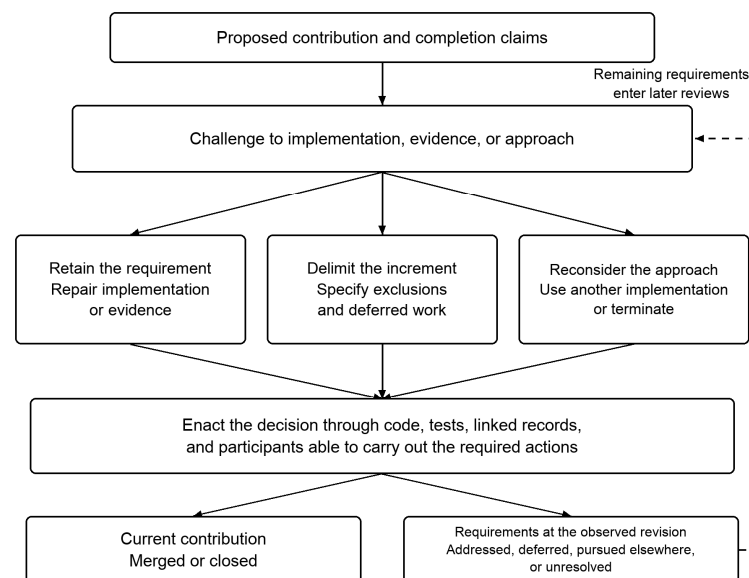


Figure 1. Contested acceptance and the continuation of requirements. The pathways can combine or follow one another. Decisions take effect through code, tests, linked records, and participants able to perform the required actions. The two outcome boxes distinguish the current contribution’s disposition from the treatment of its requirements. The dashed arrow follows requirements into later review. In the squiff trajectory, subsequent decisions both repartitioned the work and reversed changes in an earlier accepted revision. Path A is supported where the requirement remains binding and a feasible repair is enacted; path B where authorized separation of residual work is enacted; path C where the current approach loses support or an explicit alternative supersedes it. The paths express conditional process relations and may overlap. Unexplained closures remain unassigned.

The contrasts support three conditional process propositions. First, when reviewers retain a requirement as binding and participants can change its implementation or evaluation, the observed route is repair under the retained obligation (A). Primer's timer requirement and sequin's request assertion illustrate two targets of that repair; shopware shows that a change of actor can make the repair possible.

Second, when participants can separate a useful increment from residual work and authorize that separation, explicit exclusions and linked records can make acceptance possible while the residual requirement remains (B). Experimental dialect support in squuff provides the clearest instruction-to-configuration sequence. Git-link supplies a partial comparison because its scope instruction is visible but the removal patch is not independently established in the selected evidence.

Third, when the implementation conflicts with consequential project dependencies, or an explicit alternative or project change displaces it, the current approach may be redirected or terminated (C). The shopware cookie approach, the CV07 replacement, and the bun rewrite illustrate different grounds for that transition. Across the paths, later revisions can restore, defer again, or reverse the treatment of the same requirement, as the squuff test trajectory demonstrates.

5. Discussion

5.1. Completion as a Relation Between Accepted and Remaining Work

An accepted increment acquires meaning through the requirements enacted in its tests, code, and work records. Following residual requirements reveals how this local completion contributes to an undertaking that continues beyond integration.

This interpretation develops the connection between delegation and the work of completing a delegated contribution. Delegation theory already accommodates ambiguity and ongoing coordination [3,4]. The present material specifies one object of that coordination: the relation between an acceptance decision and the requirements that remain after it. The temporal significance of this relation becomes visible when a deferred requirement reappears in a later contribution and is then repaired or partitioned again. A completion decision can thus generate the conditions for another episode of work while still being a consequential endpoint for the current increment.

Acceptance also differs from assigning responsibility for further work. An issue records an unresolved requirement, whereas a participant's commitment identifies an accountable actor; the first can exist without the second. A replacement can similarly continue a task through another contributor. These relations make it possible to examine how recorded work becomes actionable.

This extends the account of how practice objects become stabilized [26] by following both the accepted increment and its residual work. Issue closure alongside continuing test exclusions shows that stabilization is local and temporal: the record of acceptance and the requirement's technical state can diverge.

Test-maintenance research establishes restoration, renewed exclusion, and divergence from issue status [30]; technical-debt research shows how review records unfinished work [29]. The present comparison connects those histories to the authorization that makes an increment acceptable. Ordinary incremental development explains staged delivery. Tracing who permits an omission, which evaluation artifact implements it, and how later versions change it explains the content and reversibility of a particular stage. In primer and squuff, similar test exclusions carried different obligations in the migration and experimental-extension tasks. Their significance becomes visible in the enacted decision and the requirement's later treatment.

5.2. Feedback Can Change the Criterion and the Means of Evaluation

A systems view makes the consequences of these decisions easier to specify. Review connects execution to evaluation through feedback [43]. Some feedback requests corrective action to meet an established requirement, as in the instruction to fix timer tests. Other feedback changes the scope of the current increment, as in the authorized T-SQL exclusions. Still other feedback repairs the evaluative apparatus, as in the test that could not distinguish whether a request had been sent. The system's response therefore depends on what the feedback addresses and what participants are able to modify.

This differentiation is relevant to sociotechnical systems engineering, which treats organizational and technical arrangements as interdependent features of system design and operation [1,44]. In *sqruff*, an ignored-test marker implements an authorized deferral in the executable configuration: the test is marked to be skipped while its associated requirement is recorded for further work. A reviewer comment can authorize that configuration, while a linked issue can preserve its implications for later work. The acceptance decision emerges through these connected elements. The relation between the instruction and the configuration gives the decision its practical meaning.

The sequin episode further specifies how evidence enters this system. Once a participant shows that a test can pass without the expected behavior, a passing result can no longer establish that the expected behavior occurred. The revised assertion changes the connection between an observation and the behavioral claim being judged. This extends the situated examination of AI-related knowledge claims into the construction of the testing artifact itself [32]. Accounts of passing-test heuristics identify a vulnerability in agent oversight [35]. The sequin sequence shows how that vulnerability can be addressed by exposing a failure of discrimination and changing the observation required for local acceptance. When implementation and proposed tests enter review together, the test's adequacy becomes part of the work required for acceptance.

This gives practical content to verifiability in delegation design [8]. A stated criterion and an associated test are insufficient if the test cannot establish whether the criterion has been met. Supporting verification requires maintaining the connection among the requirement, its operational test, and interpretation of the result as work changes.

5.3. Implications for Research and the Design of Review Systems

Earlier studies establish changing requirements, replacement, and heterogeneous reasons for disposition [5,15–19]. Following identifiable requirements specifies what continues across those categories: the behavior carried into another artifact, the evaluation attached to it, and later changes to its acceptance conditions. This connection between requirements and later revisions constitutes the process contribution.

Human–agent testing comparisons establish both continuities and differences with human work [20]. The cases here specify how execution arrangements shape continuation. Primer's interface restriction required human action to open a replacement; disabling the integration in shopware shifted revision to the maintainer. These documented transitions locate agency in the people, interfaces, and permissions enabling the next action. Subsequent human commits remain distinct from autonomous agent capability.

Outcome measures should reflect these distinctions. Merge status measures integration of an original contribution; assessing task accomplishment also requires attention to accepted scope, acknowledged exclusions, and linked work. Replacement differs from abandonment, and integration with unsupported behavior differs from demonstrated satisfaction. Following these relations complements classifications of rejection reasons [13,14].

For review-system design, the cases support making the connection between scope and evidence clear when an integration decision is made. A review summary could identify the

behavior being accepted, the test or observation supporting that claim, and any requirement assigned to a linked artifact. Such a representation would help a later contributor interpret why a test was disabled or why an issue closed [12,25]. Its purpose is to preserve the meaning of a decision across time. Merely adding another completed checkbox would reproduce the ambiguity identified in the *sqruff* continuation.

Preserving the meaning of a decision also requires a feasible route for acting on it. In the *primer* and *shopware* episodes, the requested continuation involved moving the work to a new PR or stopping an integration. Review tools can support this by distinguishing a requested code revision from an action requiring project permissions. This design implication is consistent with support for interdependence in Coactive Design [39]. In the present setting, its value is concrete: an actionable request must connect the disputed requirement to a participant and artifact through which the next change can occur.

5.4. Scope and Limitations

The process model is analytically applicable where modular contributions, revisable evaluation artifacts, and local acceptance decisions make requirements traceable across versions. Under these conditions, the relation between a stated obligation, its enactment, and later treatment can be examined beyond the original PR. The strongest comparisons concern identifiable requirements linked by explicit task relations or code and test identifiers. These comparisons connect changing requirements to the actions and artifacts through which work continues.

The study relies on public GitHub records and cannot recover private discussions, undocumented decisions, or intentions beyond participants' stated accounts. Timeline coverage is incomplete for many cases, and absence of a retrieved event does not establish absence of activity. Reports of passing tests do not replace independent execution evidence; source labels and commit accounts do not fully establish how the code was produced. Eight unmerged cases have no explicit closing reason, preventing attribution of those endpoints to a pathway. Purposive selection used contestation and known dispositions, so the 30 cases do not estimate pathway prevalence or the causal effect of negotiation on acceptance. The frame screening describes visible contestation under its own definition and cannot be equated with studies measuring missing rejection explanations. Without a non-agentic comparison, the analysis cannot establish that the observed mechanisms are unique to agents or more common in agent-associated work.

6. Conclusions

These cases connect conditions imposed in review to their enactment in code, tests, and work records. Using acceptance boundary as a sensitizing distinction, this analysis explains conditional, overlapping paths of repair, authorized deferral, and redirection. *Sqruff*'s partial restoration and subsequent rollback show how accepted arrangements remain revisable. This process account connects local integration to the changing requirements of the larger undertaking and locates agent-associated completion in the contributions of people, tools, and evaluation artifacts.

Supplementary Materials: The following supporting information can be downloaded at: <https://www.mdpi.com/article/10.3390/systems14101312/s1>, Supplementary Material S1 provides case evidence and linked artifacts; S2 provides evidence entry points, selection rules, and manual coding; S3 traces the *sqruff* requirements through repair and rollback. Table S1. Event anchors for *primer/react* #6416; Table S2. Event anchors for *Azure/bicep-registry-modules* #5692; Table S3. Event anchors for *shopware/shopware* #8383; Table S4. Event anchors for *sshaw/git-link* #144; Table S5. Event anchors for *AliAkhtari78/SpotifyScraper* #21; Table S6. Event anchors for *dotnet/runtime* #115762; Table S7. Event anchors for *dotnet/runtime* #115826; Table S8. Event anchors for

dotnet/fsharp #18583; Table S9. Event anchors for near/nearcore #13318; Table S10. Event anchors for Arize-ai/phoenix #8660; Table S11. Event anchors for microsoft/finops-toolkit #1621; Table S12. Event anchors for microsoft/typespec #7619; Table S13. Event anchors for microsoft/typescript-go #1387; Table S14. Event anchors for specmatic/specmatic #1951; Table S15. Event anchors for openai/codex #1601; Table S16. Event anchors for microsoft/onnxscript #2455; Table S17. Event anchors for airbytehq/airbyte #52636; Table S18. Event anchors for dotCMS/core #32657; Table S19. Event anchors for quarylabs/sqruff #1664; Table S20. Event anchors for quarylabs/sqruff #1706; Table S21. PR-level evidence and public-discussion coverage for mitsuhiro/insta #750; Table S22. Event anchors for openai/codex #1547; Table S23. Event anchors for openai/codex #1541; Table S24. Event anchors for openai/codex #1545; Table S25. Event anchors for openai/codex #1542; Table S26. Event anchors for sequinstream/sequin #1795; Table S27. Event anchors for oven-sh/bun #20298; Table S28. Event anchors for oven-sh/bun #20736; Table S29. Event anchors for liam-hq/liam #1380; Table S30. Event anchors for shopware/shopware #8269; Table S31. Version and disposition anchors; Table S32. Evidence entry points for the cases interpreted in manuscript Table 3; Table S33. Initial independent screening judgments (n = 535); Table S34. Requirement trajectories in sqruff.

Author Contributions: Conceptualization, M.L.; methodology, M.L., J.Z. and X.L.; software, M.L.; validation, M.L., J.Z. and X.L.; formal analysis, M.L.; investigation, M.L.; resources, J.Z.; data curation, M.L.; writing—original draft preparation, M.L.; writing—review and editing, M.L., J.Z. and X.L.; visualization, M.L.; supervision, J.Z.; project administration, M.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Postgraduate Research & Practice Innovation Program of Jiangsu Province, grant number 26CXJY3027.

Institutional Review Board Statement: This study was exempt from ethical review under Article 32(1) of China's Measures for the Ethical Review of Life Science and Medical Research Involving Humans (2023), which provides exemption for research using lawfully obtained public data or data generated through observation without interference with public behavior, provided that the research causes no harm to the human body and involves neither sensitive personal information nor commercial interests. It used publicly available GitHub records for secondary analysis and involved no participant recruitment or contact, human or animal experiments, or biological materials. Public records were summarized for process analysis, with selected excerpts and source links retained to support the findings.

Informed Consent Statement: Not applicable. The study analyzed publicly available records and involved no participant recruitment or direct interaction.

Data Availability Statement: The original source dataset is publicly available as AIDev, version 3 [41]. The GitHub PRs examined in the study are identified and linked in Appendix A, and selected public evidence is indexed in Supplementary Material S1. Public evidence is de-identified in its presentation by replacing personal account names with participant roles. S1 retains event identifiers, timestamps, and source links for evidence verification. The larger compiled cross-repository event database and analytical working files are available from the corresponding author upon reasonable request, subject to privacy safeguards and applicable source licenses. GitHub source records remain publicly accessible and may be modified by their respective projects.

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

Appendix A. Diagnostic PRs

Table A1. Diagnostic case set. Agent names identify the source classification of each contribution. “Closed” means closed without merging for the original PR; linked continuation is considered separately in the analysis.

Repository and PR	Agent Association	Original PR Outcome	Decision Date (UTC)
primer/react #6416 (https://github.com/primer/react/pull/6416) (accessed on 15 July 2026)	Copilot	Closed	1 August 2025
Azure/bicep-registry-modules #5692 (https://github.com/Azure/bicep-registry-modules/pull/5692) (accessed on 15 July 2026)	Copilot	Closed	24 July 2025
shopware/shopware #8383 (https://github.com/shopware/shopware/pull/8383) (accessed on 15 July 2026)	Devin	Merged	16 April 2025
sshaw/git-link #144 (https://github.com/sshaw/git-link/pull/144) (accessed on 15 July 2026)	Copilot	Merged	12 July 2025
AliAkhtari78/SpotifyScraper #21 (https://github.com/AliAkhtari78/SpotifyScraper/pull/21) (accessed on 15 July 2026)	Copilot	Merged	22 May 2025
dotnet/runtime #115762 (https://github.com/dotnet/runtime/pull/115762) (accessed on 15 July 2026)	Copilot	Closed	22 May 2025
dotnet/runtime #115826 (https://github.com/dotnet/runtime/pull/115826) (accessed on 15 July 2026)	Copilot	Closed	7 September 2025
dotnet/fsharp #18583 (https://github.com/dotnet/fsharp/pull/18583) (accessed on 15 July 2026)	Copilot	Closed	30 July 2025
near/nearcore #13318 (https://github.com/near/nearcore/pull/13318) (accessed on 15 July 2026)	Devin	Merged	8 April 2025
Arize-ai/phoenix #8660 (https://github.com/Arize-ai/phoenix/pull/8660) (accessed on 15 July 2026)	Cursor	Merged	21 July 2025
microsoft/finops-toolkit #1621 (https://github.com/microsoft/finops-toolkit/pull/1621) (accessed on 15 July 2026)	Copilot	Merged	22 May 2025
microsoft/typespec #7619 (https://github.com/microsoft/typespec/pull/7619) (accessed on 15 July 2026)	Copilot	Closed	12 June 2025
microsoft/typescript-go #1387 (https://github.com/microsoft/typescript-go/pull/1387) (accessed on 15 July 2026)	Copilot	Closed	16 September 2025
specmatic/specmatic #1951 (https://github.com/specmatic/specmatic/pull/1951) (accessed on 15 July 2026)	Copilot	Closed	17 September 2025
openai/codex #1601 (https://github.com/openai/codex/pull/1601) (accessed on 15 July 2026)	Codex	Merged	18 July 2025
microsoft/onnxsript #2455 (https://github.com/microsoft/onnxsript/pull/2455) (accessed on 15 July 2026)	Copilot	Merged	15 July 2025
airbytehq/airbyte #52636 (https://github.com/airbytehq/airbyte/pull/52636) (accessed on 15 July 2026)	Devin	Merged	11 February 2025
dotCMS/core #32657 (https://github.com/dotCMS/core/pull/32657) (accessed on 15 July 2026)	Claude Code	Merged	20 March 2026
quarylabs/sqruff #1664 (https://github.com/quarylabs/sqruff/pull/1664) (accessed on 15 July 2026)	Claude Code	Merged	23 June 2025

Table A1. Cont.

Repository and PR	Agent Association	Original PR Outcome	Decision Date (UTC)
quarylabs/sqruff #1706 (https://github.com/quarylabs/sqruff/pull/1706) (accessed on 15 July 2026)	Claude Code	Closed	29 June 2025
mitsuhiko/insta #750 (https://github.com/mitsuhiko/insta/pull/750) (accessed on 15 July 2026)	Claude Code	Merged	2 April 2025
openai/codex #1547 (https://github.com/openai/codex/pull/1547) (accessed on 15 July 2026)	Codex	Closed	17 July 2025
openai/codex #1541 (https://github.com/openai/codex/pull/1541) (accessed on 15 July 2026)	Codex	Merged	14 July 2025
openai/codex #1545 (https://github.com/openai/codex/pull/1545) (accessed on 15 July 2026)	Codex	Closed	17 July 2025
openai/codex #1542 (https://github.com/openai/codex/pull/1542) (accessed on 15 July 2026)	Codex	Merged	13 July 2025
sequinstream/sequin #1795 (https://github.com/sequinstream/sequin/pull/1795) (accessed on 15 July 2026)	Codex	Merged	16 June 2025
oven-sh/bun #20298 (https://github.com/oven-sh/bun/pull/20298) (accessed on 15 July 2026)	Cursor	Closed	26 June 2026
oven-sh/bun #20736 (https://github.com/oven-sh/bun/pull/20736) (accessed on 15 July 2026)	Cursor	Closed	26 June 2026
liam-hq/liam #1380 (https://github.com/liam-hq/liam/pull/1380) (accessed on 15 July 2026)	Devin	Closed	24 April 2025
shopware/shopware #8269 (https://github.com/shopware/shopware/pull/8269) (accessed on 15 July 2026)	Devin	Closed	12 May 2025

References

- Baxter, G.; Sommerville, I. Socio-technical systems: From design methods to systems engineering. *Interact. Comput.* **2011**, *23*, 4–17. [\[CrossRef\]](#)
- Di Maio, P. Towards a Metamodel to Support the Joint Optimization of Socio Technical Systems. *Systems* **2014**, *2*, 273–296. [\[CrossRef\]](#)
- Baird, A.; Maruping, L.M. The Next Generation of Research on IS Use: A Theoretical Framework of Delegation to and from Agentic IS Artifacts. *MIS Q.* **2021**, *45*, 315–341. [\[CrossRef\]](#)
- Stelmaszak, M.; Möhlmann, M.; Sørensen, C. When Algorithms Delegate to Humans: Exploring Human-Algorithm Interaction at Uber. *MIS Q.* **2025**, *49*, 305–330. [\[CrossRef\]](#)
- Tsay, J.; Dabbish, L.; Herbsleb, J. Let's talk about it: Evaluating contributions through discussion in GitHub. In Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering, Hong Kong, China, 16–21 November 2014; pp. 144–154. [\[CrossRef\]](#)
- Steinmacher, I.; Pinto, G.; Wiese, I.S.; Gerosa, M.A. Almost There: A Study on Quasi-Contributors in Open Source Software Projects. In Proceedings of the 40th International Conference on Software Engineering, Gothenburg, Sweden, 27 May–3 June 2018; pp. 256–266. [\[CrossRef\]](#)
- Fügener, A.; Grahl, J.; Gupta, A.; Ketter, W. Cognitive Challenges in Human–Artificial Intelligence Collaboration: Investigating the Path Toward Productive Delegation. *Inf. Syst. Res.* **2022**, *33*, 678–696. [\[CrossRef\]](#)
- Tomašev, N.; Franklin, M.; Osindero, S. Intelligent AI Delegation. *arXiv* **2026**, arXiv:2602.11865. [\[CrossRef\]](#)
- Dai, C.P.; Yang, M.; Lee, S. Designing Human–AI Collaboration for Hybrid Intelligence in Immersive Learning Environments: A Conceptual Framework. *Systems* **2026**, *14*, 639. [\[CrossRef\]](#)
- Seeber, I.; Bittner, E.; Briggs, R.O.; de Vreede, T.; de Vreede, G.J.; Elkins, A.; Maier, R.; Merz, A.B.; Oeste-Reiß, S.; Randrup, N.; et al. Machines as teammates: A research agenda on AI in team collaboration. *Inf. Manag.* **2020**, *57*, 103174. [\[CrossRef\]](#)
- Bao, Y.; Gong, W.; Yang, K. A Literature Review of Human–AI Synergy in Decision Making: From the Perspective of Affordance Actualization Theory. *Systems* **2023**, *11*, 442. [\[CrossRef\]](#)

12. Bacchelli, A.; Bird, C. Expectations, outcomes, and challenges of modern code review. In Proceedings of the 2013 35th International Conference on Software Engineering (ICSE), San Francisco, CA, USA, 18–26 May 2013; pp. 712–721. [\[CrossRef\]](#)
13. Abujadallah, M.; Arabat, A.; Sayagh, M. Understanding the Rejection of Fixes Generated by Agentic Pull Requests—Insights from the AIDev Dataset. In Proceedings of the 23rd International Conference on Mining Software Repositories, Rio de Janeiro, Brazil, 13–14 April 2026; pp. 873–877. [\[CrossRef\]](#)
14. Nakashima, S.; Ishimoto, Y.; Kondo, M.; McIntosh, S.; Kamei, Y. Why Agentic-PRs Get Rejected: A Comparative Study of Coding Agents. *arXiv* **2026**, arXiv:2602.04226. [\[CrossRef\]](#)
15. Ogenrwot, D.; Businge, J. PatchTrack: A comprehensive analysis of ChatGPT’s influence on pull request outcomes. *Empir. Softw. Eng.* **2026**, *31*, 136. [\[CrossRef\]](#)
16. Ehsani, R.; Pathak, S.; Rawal, S.; Al Mujahid, A.; Imran, M.M.; Chatterjee, P. Where Do AI Coding Agents Fail? An Empirical Study of Failed Agentic Pull Requests in GitHub. In Proceedings of the 23rd International Conference on Mining Software Repositories, Rio de Janeiro, Brazil, 13–14 April 2026; pp. 807–811. [\[CrossRef\]](#)
17. Hindi, M.; Mahmood, Y.; Mohammed, L.; Bouktif, S.; Mediani, M. Coding Agents in the Wild: Failure Modes and Rejection Patterns of AI-Generated Pull Requests. *IEEE Access* **2026**, *14*, 83075–83094. [\[CrossRef\]](#)
18. Strauss, M.C.; Schulze, S. Why and When Agentic Pull Requests are (not) Accepted: An Exploratory Study. In Proceedings of the 23rd International Conference on Mining Software Repositories, Rio de Janeiro, Brazil, 13–14 April 2026; pp. 767–771. [\[CrossRef\]](#)
19. Peralta, S.R.O.; Hoshi, F.; Washizaki, H.; Ubayashi, N.; Kondo, I.; Higo, Y.; Mukai, H.; Yoshida, N.; Kusama, K.; Tanaka, H.; et al. Why Are Agentic Pull Requests Merged or Rejected? An Empirical Study. In Proceedings of the 23rd International Conference on Mining Software Repositories, Rio de Janeiro, Brazil, 13–14 April 2026; pp. 787–791. [\[CrossRef\]](#)
20. Milanese, R.; Salzano, F.; Spina, A.; Vitale, A.; Pareschi, R.; Fasano, F.; Fazzini, M. Human-Agent versus Human Pull Requests: A Testing-Focused Characterization and Comparison. In Proceedings of the 23rd International Conference on Mining Software Repositories, Rio de Janeiro, Brazil, 13–14 April 2026; pp. 999–1003. [\[CrossRef\]](#)
21. Strauss, A. The Articulation of Project Work: An Organizational Process. *Sociol. Q.* **1988**, *29*, 163–178. [\[CrossRef\]](#)
22. Schmidt, K.; Bannon, L. Taking CSCW Seriously: Supporting Articulation Work. *Comput. Support. Coop. Work (CSCW)* **1992**, *1*, 7–40. [\[CrossRef\]](#)
23. Schmidt, K.; Simone, C. Coordination mechanisms: Towards a conceptual foundation of CSCW systems design. *Comput. Support. Coop. Work (CSCW)* **1996**, *5*, 155–200. [\[CrossRef\]](#)
24. Gasser, L. The integration of computing and routine work. *ACM Trans. Inf. Syst.* **1986**, *4*, 205–225. [\[CrossRef\]](#)
25. Suchman, L. Making work visible. *Commun. ACM* **1995**, *38*, 56–64. [\[CrossRef\]](#)
26. Allen, D.; May, C. Organizing Practice and Practicing Organization: An Outline of Translational Mobilization Theory. *Sage Open* **2017**, *7*, 2158244017707993. [\[CrossRef\]](#)
27. Carlile, P.R. Transferring, Translating, and Transforming: An Integrative Framework for Managing Knowledge Across Boundaries. *Organ. Sci.* **2004**, *15*, 555–568. [\[CrossRef\]](#)
28. Star, S.L.; Griesemer, J.R. Institutional Ecology, ‘Translations’ and Boundary Objects: Amateurs and Professionals in Berkeley’s Museum of Vertebrate Zoology, 1907–39. *Soc. Stud. Sci.* **1989**, *19*, 387–420. [\[CrossRef\]](#)
29. Kashiwa, Y.; Nishikawa, R.; Kamei, Y.; Kondo, M.; Shihab, E.; Sato, R.; Ubayashi, N. An empirical study on self-admitted technical debt in modern code review. *Inf. Softw. Technol.* **2022**, *146*, 106855. [\[CrossRef\]](#)
30. Kim, D.J.; Yang, B.; Yang, J.; Chen, T.-H. How disabled tests manifest in test maintainability challenges? In Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Athens, Greece, 23–28 August 2021; pp. 1045–1055. [\[CrossRef\]](#)
31. Passi, S.; Jackson, S.J. Trust in Data Science: Collaboration, Translation, and Accountability in Corporate Data Science Projects. *Proc. ACM Hum.-Comput. Interact.* **2018**, *2*, 136. [\[CrossRef\]](#)
32. Lebovitz, S.; Lifshitz-Assaf, H.; Levina, N. To Engage or Not to Engage with AI for Critical Judgments: How Professionals Deal with Opacity When Using AI for Medical Diagnosis. *Organ. Sci.* **2022**, *33*, 126–148. [\[CrossRef\]](#)
33. Pachidi, S.; Berends, H.; Faraj, S.; Huysman, M. Make Way for the Algorithms: Symbolic Actions and Change in a Regime of Knowing. *Organ. Sci.* **2021**, *32*, 18–41. [\[CrossRef\]](#)
34. Faraj, S.; Pachidi, S.; Sayegh, K. Working and organizing in the age of the learning algorithm. *Inf. Organ.* **2018**, *28*, 62–70. [\[CrossRef\]](#)
35. Dhanorkar, S.; Passi, S.; Vorvoreanu, M. Human oversight of agentic systems in practice: Examining the oversight work, challenges, and heuristics of developers using software agents. In Proceedings of the 2026 ACM Conference on Fairness, Accountability, and Transparency, Montreal, QC, Canada, 25–28 June 2026; pp. 6438–6465. [\[CrossRef\]](#)
36. Raisch, S.; Krakowski, S. Artificial Intelligence and Management: The Automation–Augmentation Paradox. *Acad. Manag. Rev.* **2021**, *46*, 192–210. [\[CrossRef\]](#)
37. Bainbridge, L. Ironies of automation. *Automatica* **1983**, *19*, 775–779. [\[CrossRef\]](#)
38. Simkute, A.; Tankelevitch, L.; Kewenig, V.; Scott, A.E.; Sellen, A.; Rintel, S. Ironies of Generative AI: Understanding and Mitigating Productivity Loss in Human-AI Interaction. *Int. J. Human–Computer Interact.* **2025**, *41*, 2898–2919. [\[CrossRef\]](#)

39. Johnson, M.; Bradshaw, J.M.; Feltovich, P.J.; Jonker, C.M.; Van Riemsdijk, M.B.; Sierhuis, M. Coactive Design: Designing Support for Interdependence in Joint Activity. *J. Hum.-Robot Interact.* **2014**, *3*, 43–69. [[CrossRef](#)]
40. Ling, M.; Zou, J. After the lights go out: Announcement-phase public relational expression and later continuity-oriented responses to an AI agent shutdown in an account-linked longitudinal study of Weibo posts. *Front. Psychol.* **2026**, *17*, 1971726. [[CrossRef](#)] [[PubMed](#)]
41. Li, H.; Zhang, H.; Hassan, A.E. AIDev: Studying AI Coding Agents on GitHub. *Zenodo* **2025**. Dataset, version 3. [[CrossRef](#)]
42. Langley, A. Strategies for Theorizing from Process Data. *Acad. Manag. Rev.* **1999**, *24*, 691–710. [[CrossRef](#)] [[PubMed](#)]
43. Sadowski, C.; Söderberg, E.; Church, L.; Sipko, M.; Bacchelli, A. Modern Code Review: A Case Study at Google. In Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice, Gothenburg, Sweden, 27 May–3 June 2018; pp. 181–190. [[CrossRef](#)]
44. Gregoriades, A.; Sutcliffe, A. Using Task Support Requirements during Socio-Technical Systems Design. *Systems* **2024**, *12*, 348. [[CrossRef](#)]

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