International Journal of Modern Computer Science and IT Innovations (IJMCSIT)
eISSN: 3087-4289
Volume03 Issue10 | Page No: - 42-55

Received: 24 Aug 2026, Revised: 15 Sep 2026, Accepted: 30 Sep 2026, Published: 23 Oct 2026

DOI: https://doi.org/10.55640/ijmcsit-v03i10-04

Micro-Frontend And Microservices Architectures In Bond ETF Workflow Management: A Narrative Literature Review And Gap Analysis

Tejas Patil
Independent Researcher, USA
Email: tejasarvindpatil1@gmail.com

ABSTRACT: This paper reviews the software-architecture and market-structure literature relevant to managing bond exchange-traded fund (ETF) creation and redemption workflows within enterprise asset management platforms and identifies gaps that the reviewed sources do not address. Bond ETF creation and redemption is operationally distinct from equity ETF mechanics because fixed-income index baskets are too large to replicate exactly, so custom baskets are constructed for each transaction, and this review takes them to be negotiated between authorized participants and fund portfolio managers under same-day net asset value and reporting constraints. Two software-decomposition patterns bear on how such workflows are built: micro-frontend (micro-UI) architecture on the presentation tier, and microservices with event-driven coordination on the backend. Both are now well represented in the academic record, and peer-reviewed literature documents their application across general enterprise and industrial software contexts. This review draws on peer-reviewed and indexed academic sources, together with the United States rule governing exchange-traded fund baskets, to characterize each area. These include systematic mapping studies and systematic literature reviews on micro-frontends and microservices, foundational microservices literature, empirical case studies of strangler-fig-based migration, and finance journal research on bond ETF creation, redemption, and liquidity. The synthesis finds that, while each literature is individually substantial, none of the reviewed sources jointly addresses the specific operational shape of bond ETF workflow management: a multi-round, multi-party basket negotiation; dual cash and in-kind settlement paths that must reconcile to one fund position; and role-differentiated approval steps operating under hard same-day deadlines for net asset value calculation and reporting. Multi-round basket negotiation between those parties is an assumption defining the review's scope and is not drawn from the cited sources. The paper does not propose or evaluate a new architecture. Its contribution is to locate this gap, as found among the sources reviewed, across four literature threads and to set out what a future design-oriented study would need to address.

KEYWORDS: Micro-Frontend Architecture; Microservices; Event-Driven Architecture; Bond Exchange-Traded Funds; ETF Creation and Redemption; Legacy System Modernization; Fixed-Income Market Structure; Narrative Literature Review

1. Introduction

The mechanism that keeps a bond exchange-traded fund (ETF) trading close to its net asset value (NAV), known as creation and redemption, is operationally more complex for fixed-income funds than for equity funds. An equity ETF basket is typically a close, mechanical replica of the fund's holdings. A fixed-income index, by contrast, spans many bonds of uneven liquidity. Koont, Ma, Pástor, and Zeng (2025) show that corporate bond ETFs choose creation and redemption baskets containing cash and only a subset of index bonds, and in practice the basket is

negotiated between an authorized participant (AP) and the fund's portfolio manager instead of being generated automatically. This negotiation, together with the compliance checks, cash and in-kind allocation, trade booking, and settlement and NAV reconciliation steps that follow it, constitutes an operationally demanding workflow. Enterprise asset management platforms must support it on same-day timelines and for several internal roles. Throughout this paper, asset management refers to investment management, meaning the management of funds and portfolios, and not to the separate software category of enterprise asset management concerned with maintaining physical equipment.

Independently of this market-structure problem, enterprise software architecture has undergone two related decompositions over roughly the past decade. On the backend, monolithic order-management and portfolio systems have progressively been broken into microservices organized around business capabilities, frequently coordinated through asynchronous, event-driven messaging in place of synchronous request-response calls. On the frontend, an analogous pattern known as micro-frontend or micro-UI architecture allows independently built, tested, and deployed user-interface modules to compose into one coherent application. Both patterns are documented in peer-reviewed software-engineering literature, including systematic mapping studies and systematic literature reviews. That literature, however, has been developed almost entirely outside the financial-services domain.

This paper's central research problem can be stated as a single question: how adequately do existing micro-frontend and microservices architectures support the distinctive workflow characteristics of bond ETF creation and redemption? The central claim is that the two decomposition literatures, together with the finance literature describing how bond ETF creation and redemption works, have not yet been examined jointly. The gap matters because the domain does not map cleanly onto the general case studies that dominate this literature. The basket negotiation between the AP and the portfolio manager is a multi-round, multi-party process rather than a single triggering event. It must support several distinct user roles interacting with the same evolving object. And its two settlement paths, cash and in-kind, must ultimately reconcile to one fund position under a hard, same-day NAV cutoff. No source identified in this review addresses this combination directly, in either the software-architecture or the finance literature, and these three workflow characteristics are adopted as scope assumptions, set out in Section 3.5.

Although bond ETF creation and redemption is the specific and primary domain of this review, the workflow characteristics that motivate it are shared, in part, by other negotiated and regulated financial processes, such as primary issuance of fixed-income securities, bilaterally negotiated derivatives confirmation, and collateral substitution in securities lending. Each of these involves multi-party negotiation of an evolving instruction under regulatory time constraints. The gaps identified here for bond ETF workflows may therefore have wider relevance to that broader class of negotiated financial workflows, while bond ETF creation and redemption remains the concrete focus throughout.

1.1 Significance and Novel Aspect of This Work

The novel aspect of this paper is a gap analysis at the intersection of three primary literatures that have so far been reviewed separately: micro-frontend architecture, microservices and event-driven backend design, and bond ETF market structure. A fourth thread, legacy modernization, is reviewed in a supporting role because it governs how any target architecture would be reached in practice. Prior reviews of micro-frontend architecture, such as the multivocal literature review by Peltonen, Mezzalira, and Taibi (2021) and the systematic mapping study by Cunha de Amorim and Canedo (2025), synthesize the general software-engineering literature on the pattern but do not address financial-services workflows or the ETF creation and redemption problem.

Broad microservices reviews likewise treat the pattern generically: Li, Zhang, and colleagues (2021) catalog quality attributes across the microservices literature, and Abgaz et al. (2023) review how monoliths are decomposed into services, but neither is oriented to a specific financial workflow. Prior finance literature on bond ETF creation and redemption, such as Finnerty, Reisel, and Zhong (2025) in the Journal of Financial and Quantitative Analysis, treats the mechanism as a market structure and liquidity question, not as a software-systems design problem. This paper's contribution is to place these bodies of work side by side, under a single set of review questions, and to characterize where they fail to meet.

1.2 Contributions of This Review

This review makes two main contributions. The first is a cross-domain synthesis of micro-frontend architecture, microservices with event-driven backend design, and bond ETF creation and redemption, three bodies of work that have not, to the best of the review's search, been examined together. The second is a structured gap analysis derived from that synthesis, set out in Section 4 as five numbered gaps and stated as specific research questions rather than a proposed solution, and intended to motivate future design-oriented work. Supporting these, the paper also reviews the strangler fig legacy-migration pattern as the practical bridge from existing monolithic platforms to any future decomposed architecture, and it is explicit about the boundary of its own evidence: what each cited source does and does not establish.

1.3 Scope and Organization

This paper is a narrative literature review and gap analysis. Section 2 presents the literature review across the four constituent threads, closing each with what that thread does not establish. Section 3 describes the review methodology, including the review questions, search strategy, inclusion criteria, nature of the evidence, and workflow assumptions, and Figure 1 summarizes the review workflow. Section 4 presents the results of the synthesis through a comparative summary table, a coverage matrix, a comparative discussion, and five numbered gaps. Section 5 concludes with limitations and future directions, each traced to a gap.

2. Literature Review

2.1 Micro-Frontend (Micro-UI) Architecture

Micro-frontend architecture extends the microservices decomposition principle to the presentation tier. Taibi and Mezzalira (2022) describe the approach as decomposing the frontend into individual and semi-independent micro applications, which allows a team to own a vertical slice of functionality from the database through the backend to the interface, and they list adopting organizations including SAP, Springer, Zalando, New Relic, IKEA, Starbucks, and DAZN. The list spans enterprise software, publishing, retail, and media. It contains no asset manager, bank, or other financial institution, which suggests that the pattern's documented industrial evidence base lies outside financial services.

Peltonen, Mezzalira, and Taibi's (2021) multivocal literature review, published in Information and Software Technology, surveyed 173 academic and grey-literature sources and narrowed them to 43 that reported motivations, benefits, and issues. As the authors note, the great majority of that final pool was grey literature rather than peer-reviewed work, which reflects how much of the early knowledge on micro-frontends originated in industry practice. The review describes the motivating problem as one of scale. As single-page applications grow in team size and feature count, a single shared codebase, build pipeline, and release train reproduces on the client side the coordination costs that microservices were designed to solve on the server side. Classifying motivations, benefits, and issues, the authors conclude that the pattern carries to the client side the team-scaling benefit that microservices provide on the server, at the price of added overhead. The overhead they report includes increased payload size, code duplication, coupling between teams, and monitoring complexity, all of which require careful planning to offset.

Cunha de Amorim and Canedo's (2025) systematic mapping study, presented at the 27th International Conference on Enterprise Information Systems, screened 279 initially identified studies down to a final set of 24 for full analysis, and reports its findings against three research questions covering architectural patterns, impacts, and adoption challenges. Their reported architectural views, ranked by how many of the 24 studies addressed each, are led by the view of an application programming interface (API) as the single entry point (20 of 24 studies), the Single-SPA view (18), and Web Components (13), with the Module Federation and App Container views tied at 7 studies each. Among architectural patterns specifically, framework choice and inter-module communication patterns were the most discussed, while patterns such as Domain-Driven Design and Application Shell appeared in a smaller minority of the reviewed studies (6 of 24 each). On impacts, the study's own tally finds Team Efficiency and Applicability as the most frequently reported positive impacts, while Operational Complexity is the most frequently reported negative impact, followed by challenges in Reusability and Operational Costs.

One composition mechanism that has drawn recent research attention is Module Federation, which allows one application (a host) to dynamically load code exposed at runtime by another application (a remote) without a shared build step. Its origin in the Webpack bundler is

established practitioner knowledge that the reviewed sources treat as background. Because this mechanism ties micro-frontend composition to a specific bundler, subsequent work has begun investigating bundler-independent approaches. Lando and Hasselbring (2025) document an implementation extending module federation to bundlers beyond Webpack, including Rspack and Vite, while adding TypeScript-based type safety across federated module boundaries. They identify three open challenges: performance degradation from request waterfalls when federated modules load sequentially, the difficulty of debugging distributed frontend systems compared to a monolith's centralized call stack, and a steep developer learning curve associated with runtime, tool-agnostic module sharing.

A subsequent industrial case study by Kojo et al. (2026), from the workshop track of the 2025 European Conference on Software Architecture (ECSA), examines micro-frontend adoption in an e-commerce application. It extends the evidence base to a single-organization industrial setting, which again lies outside financial services. Practitioner accounts, consulted during scoping but not cited, add further operational burdens: session and identity propagation across independently deployed modules, shared-dependency version management, and design-system governance to maintain interface consistency. None of the reviewed micro-frontend literature addresses financial-services workflows specifically. Nor does any of it discuss a scenario in which several distinct user roles must interact with the same evolving, multi-step business object across module boundaries, as opposed to composing largely independent pages or widgets.

2.2 Microservices and Event-Driven Architecture in Financial Systems

Much of the reviewed literature takes its framing of microservices from Jamshidi et al. (2018), who describe the style as a software and systems architecture approach that builds on established modularization principles while emphasizing strict technical boundaries between independently implemented and independently operated services. Dragoni et al. (2017), in an earlier survey, trace the emergence of the style from service-oriented computing and set out its open problems. Domain-driven design, the practice of aligning service boundaries with business capabilities instead of technical layers, is commonly treated in practice as a primary mechanism for defining these boundaries, and the strangler fig studies reviewed in Section 2.3 apply it directly. In financial-services practice, separating account management, payment processing, and compliance and risk management into distinct bounded contexts that communicate asynchronously is a familiar application of this logic, stated here as practitioner knowledge. It allows one domain's completed action to trigger downstream processing in another domain without direct coupling between them.

Beyond this foundational framing, later studies have examined the quality attributes and operational difficulties of microservice systems, including a systematic literature review of microservice quality attributes by Li, Zhang, and colleagues (2021). Practitioner evidence is more specific on operations. Waseem et al. (2021), drawing on a survey of 106 practitioners and interviews with 6, report that the complexity of microservices systems creates design, monitoring, and testing challenges for which no dedicated solutions exist. This backend finding mirrors the debugging and monitoring difficulty that the micro-frontend literature reports on the client side. It bears on a workflow such as bond ETF creation and redemption, whose steps would be distributed across several services and whose audit requirements make end-to-end traceability important.

Event-driven architecture (EDA) is the general mechanism the microservices literature associates with enabling this loose, asynchronous coupling between bounded contexts. A single triggering event, such as a trade execution or a payment initiation, can fan out to multiple independent downstream consumers, including compliance monitoring, audit logging, and regulatory reporting. Each consumer acts on the event without the publishing service needing to know who is listening. In the analysis offered here, this fan-out structure fits the settlement, compliance, and reporting obligations that follow a completed trade. The survey by Jamshidi et al. (2018) discusses independent scalability and fault containment as recurring themes associated with this style of decoupling.

What this backend-focused literature does not address is the stage that precedes a triggering event in the bond ETF context: a proposed creation or redemption basket that is iteratively revised between two counterparties, the AP and the portfolio manager, before any single, unambiguous trade execution event exists. In the

examples reviewed here, event-driven coordination is described around a single triggering event and its downstream fan-out. None of these examples models an object that is collaboratively and iteratively modified by two external parties prior to becoming an event, and the fan-out pattern as described in them presupposes that the published object is already final. This describes the cases surveyed above. It is not offered as a general limitation of event-driven architecture.

2.3 Legacy Modernization: The Strangler Fig Pattern

Because enterprise asset management platforms rarely begin from a greenfield state, the literature on migrating legacy monolithic systems toward microservice architectures bears on how any bond-ETF-specific target architecture would actually be reached. Fritzsch, Bogner, Zimmermann, and Wagner (2019) observe that clear guidance on refactoring legacy applications into microservices is difficult to find, and they classify the refactoring approaches that the literature does propose. Ma, Li, Lee, and Lee (2022) combine the strangler fig pattern with domain-driven design into a systematic migration process and evaluate it on two existing systems, DataCustodian and BBDP, extending an earlier conference study of the Green Button system by Li, Ma, and Lu (2020). In that process, domain-driven design supplies the service boundaries that the strangler fig pattern then extracts incrementally. Established practitioner knowledge adds further conditions for such migrations, namely a gateway capable of routing requests between legacy and new components, and data synchronization that avoids reintroducing tight coupling between the migrated services and the remaining monolith.

The strangler fig case studies sit within a larger evidence base on monolith decomposition. Abgaz et al. (2023), in a systematic review published in IEEE Transactions on Software Engineering, examine 35 primary studies on decomposing monolith applications into microservices and find that identifying service boundaries remains only partially automated, and that no single method reliably combines static, dynamic, and evolutionary information about the legacy system. Their review reinforces the point that boundary identification, which the strangler fig case studies treat as a project-specific craft, is an open problem in general. It also underlines why a bond ETF workflow, with its unusual combination of negotiation, dual settlement, and same-day constraints, could not simply reuse a boundary decomposition derived from some other domain.

Survey evidence adds the organizational side of migration. Knoche and Hasselbring (2019), surveying professionals in Germany, find that traditional companies adopting microservices emphasize the maintainability of their IT systems where early adopters emphasize scalability, and that performance and transactional concerns arise during software modernization. Their review of prior work also notes that most authors recommend an incremental approach to migration, with some treating a non-incremental approach as bad practice. What this literature does not provide is guidance on what the correct bounded contexts are for an asset management platform's ETF workflow specifically, or in what sequence they should be extracted given the domain's same-day NAV-calculation and reporting dependencies. A bond-ETF-specific application of the pattern would need to determine both independently, which is the subject of the fourth direction in Section 5.2.

2.4 Bond ETF Creation and Redemption as a Market-Structure Problem

The final literature thread concerns how bond ETF creation and redemption functions as a financial-market mechanism, since any target software architecture would ultimately need to support it. Dannhauser (2017), an early study of corporate bond ETFs, finds that greater ETF ownership lowers the yield spreads of the underlying bonds and reduces the participation of liquidity traders, while its effect on the liquidity of individual bonds is insignificant or negative. Finnerty, Reisel, and Zhong (2025), writing in the Journal of Financial and Quantitative Analysis, examine the relationship between bond ETF creation and redemption activity and underlying bond-market liquidity using daily creation and redemption data. Their central finding is that inclusion of a bond in a creation or redemption basket has a favorable effect on that bond's liquidity in both high-yield and investment-grade markets, an effect that persists during periods of market stress and is stronger for redemptions than for creations.

The authors attribute the improvement primarily to ETF mispricing arbitrage, while also finding that transaction costs and bond inventory management constrain how far that arbitrage mechanism can go. Koont, Ma, Pástor, and Zeng (2025) move closer to the operational process. They show that corporate bond ETFs actively manage their

portfolios, trading off index tracking against liquidity transformation, and that funds choose creation and redemption baskets containing cash and only a subset of index assets, especially when those assets are illiquid. The same study finds that ETFs adjust their baskets dynamically to correct portfolio imbalances while facilitating arbitrage, and that basket inclusion improves bond liquidity in general but worsens it during large imbalances between creations and redemptions, such as the COVID-19 crisis.

Ben-David, Franzoni, and Moussawi (2017) describe the institutional mechanics of the primary market in which these transactions occur. They define authorized participants as the small group of institutions allowed to trade directly with the fund sponsor, report that a daily creation basket gives the names and quantities of the securities needed to form a creation unit, and note that for some fixed-income ETFs the basket may contain a different combination of securities and cash from the fund's overall portfolio. Creation and redemption transactions settle on a cycle that follows the trade date. That account covers exchange-traded funds as a class, and it treats the fixed-income case as a qualification within the general description.

Rule 6c-11 under the Investment Company Act of 1940, adopted by the Securities and Exchange Commission (2019), supplies the regulatory frame within which those baskets are built. It defines a custom basket to include one composed of a non-representative selection of the fund's portfolio holdings, and also one that differs from the initial basket used in transactions on the same business day. Use of such a basket is conditioned on written policies that set parameters for its construction and acceptance, cover revisions to and deviations from those parameters, and name the titles or roles of the investment adviser's employees required to review each custom basket for compliance. The rule also requires daily posting, before regular trading opens, of the holdings on which the next net asset value calculation will rest. It states what a fund must have in place. It does not describe how any fund builds or reviews a basket in practice.

The empirical studies above treat the creation and redemption mechanism as a market process to be studied empirically for its price and liquidity effects, even where, as in Koont, Ma, Pástor, and Zeng (2025), the fund's choice of basket is itself modeled. The negotiation between an authorized participant and a portfolio manager, assumed here to involve the proposal of a custom, non-pro-rata basket, its review, a possible counter-proposal, and eventual acceptance, is a scope assumption of this review and is set out in Section 3.5. Transaction costs and inventory constraints likewise appear in this literature as limits on arbitrage, while their implications for the systems that process a basket lie outside its scope. How that negotiation should be represented, coordinated, or reconciled as an enterprise software workflow is a question this literature does not address, and does not purport to address.

3. Methodology

3.1 Review Approach

This is a narrative literature review rather than a fully registered systematic literature review (SLR). It follows the general spirit of SLR guidance for software engineering research, defining review questions before searching and reporting what could and could not be found, but it does not claim the complete procedural rigor (pre-registration, dual-reviewer screening, formal quality scoring) that a full SLR requires. This approach was chosen because the review question is cross-domain, spanning software architecture and financial market structure, a combination for which no single indexed database or fixed search protocol would provide adequate coverage on its own.

3.2 Review Questions

The review is organized around three questions, listed below and referred to throughout as RQ1 to RQ3.

- RQ1: What architectural patterns and empirical findings exist for micro-frontend decomposition in enterprise and financial-services user interfaces?
- RQ2: What architectural patterns and empirical findings exist for microservice and event-driven backend decomposition applied to trading, order management, and settlement workflows?
- RQ3: Where do the software-architecture literatures reviewed here, together with the bond ETF market-structure literature, fail to jointly address the specific operational characteristics of bond ETF creation and redemption workflow management, and what does that gap imply for future work?

3.3 Search Strategy and Inclusion Criteria

Candidate sources were identified through keyword search across major academic databases spanning software engineering and finance, supplemented by backward citation tracing from the reference lists of retrieved papers. Search terms combined the architectural concepts under review (micro-frontend, micro-UI, microservices, event-driven architecture, strangler fig, domain-driven design) with the application domain (bond ETF, exchange-traded fund, creation and redemption, authorized participant, asset management). For inclusion, a source had to be a peer-reviewed journal article, a peer-reviewed conference or workshop paper, or an indexed academic preprint, and had to bear directly on one of the literature threads defined by the review questions.

Sources used as evidence for any finding were further required to carry a digital object identifier (DOI) and to be publicly available in full, either from the publisher or as an openly accessible version of the same work. Studies available only through their published summaries are cited for context and do not support any finding. Backward citation tracing was also used to identify earlier foundational studies in each thread. Non-peer-reviewed material such as vendor documentation and practitioner commentary was consulted during scoping to orient the search but was not treated as citable evidence, in keeping with the review's aim of resting its findings on the academic record. One primary regulatory instrument is cited alongside the academic sources, namely the United States rule governing exchange-traded fund baskets, which is cited by its official identifiers for what the rule itself requires and not as empirical evidence. Because the review spans two ordinarily separate fields, breadth of coverage across the threads was prioritized over exhaustive saturation within any single one.

3.4 Nature of the Evidence

This review synthesizes findings reported by the cited studies. It does not collect or analyze primary quantitative data of its own. Where numerical results appear, for example the liquidity findings of Finnerty, Reisel, and Zhong (2025) or the study counts reported by Cunha de Amorim and Canedo (2025), they are reported as the conclusions of the cited source rather than as original analysis. As shown in Figure 1, the review workflow runs from question framing through search, screening, and thematic extraction against the review questions, to the synthesis and gap analysis presented in Section 4.

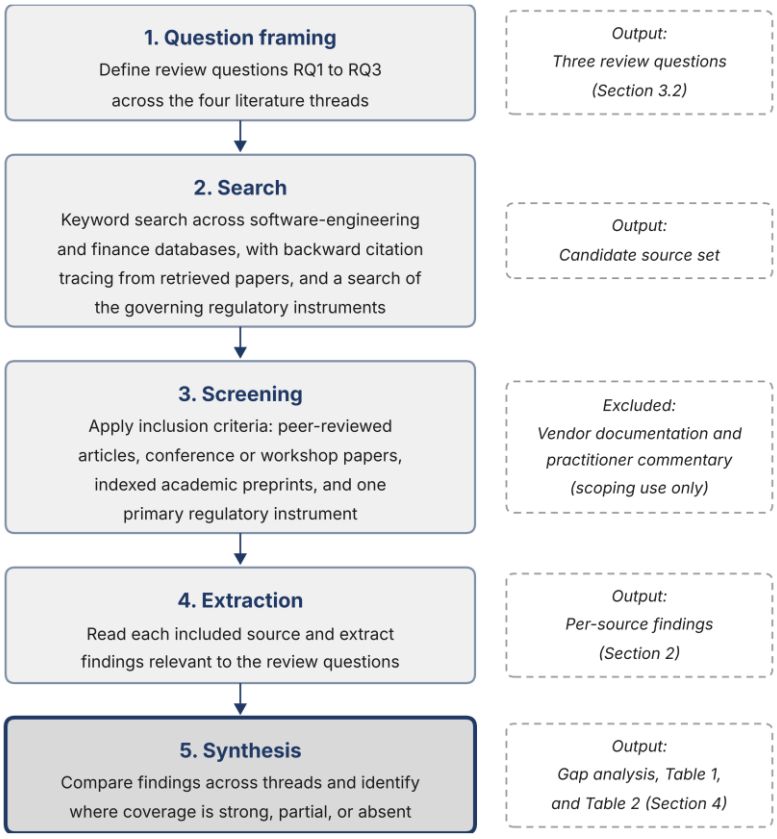


Figure 1. Review workflow, from question framing through search, screening, and thematic extraction to the synthesis and gap analysis reported in Section 4.

3.5 Workflow Assumptions Defining the Scope of the Review

The review treats four features of bond ETF workflow management as defining its scope: multi-round negotiation of custom baskets between an AP and the fund's portfolio manager, approval of those baskets by distinct internal roles, reconciliation of cash and in-kind settlement paths to one fund position, and the daily deadlines that constrain them. The deadlines are taken to be those that tie finalization of an agreed basket to that day's NAV calculation and daily reporting. Settlement is assumed to follow the trade date and is not assumed to be same-day. The order acceptance cutoff is not treated as one of these deadlines, since no source consulted here fixes it.

Three of these four features rest on cited sources. Ben-David, Franzoni, and Moussawi (2017) establish the role of the AP, the daily creation basket, baskets that mix securities and cash differently for some fixed-income funds, and a settlement cycle that follows the trade date. Rule 6c-11 conditions the use of a custom basket on written policies naming the titles or roles required to review each one, permits a basket to differ from the initial basket used on the same business day, and requires daily posting of the holdings on which the next NAV calculation will rest (Securities and Exchange Commission, 2019). The multi-round negotiation between the AP and the portfolio manager is the one feature for which the review found no source, and it is adopted as an assumption on the basis of established practice in enterprise asset management. Section 5.1 records what that costs the argument.

4. Results and Discussion

4.1 Summary of Findings by Literature Thread

Table 1 summarizes, for each of the four literature threads reviewed, the well-established findings and the extent to which that thread addresses the distinguishing properties of the bond ETF workflow. As shown in Table 1, coverage is strong within each thread and absent at their intersection.

Table 1. Summary of findings by literature thread, showing where each thread is well established and where it fails to cover bond ETF workflow specifics.

Literature Thread	Representative Source(s)	Well-Established Findings	Coverage of Bond ETF Workflow Specifics
Micro-frontend architecture	Peltonen, Mezzalira, and Taibi (2021); Taibi and Mezzalira (2022); Cunha de Amorim and Canedo (2025); Kojo et al. (2026)	API-gateway, Single-SPA, and Web Components views most reported (20, 18, 13 of 24 studies); Module Federation a mid-tier pattern (7 studies); documented costs concentrated in operational complexity, reusability, and operational costs	None found. No source addresses multi-step, stateful negotiation workflows spanning several micro-UI modules and several distinct user roles
Microservices and event-driven architecture	Dragoni et al. (2017); Jamshidi et al. (2018); Waseem et al. (2021)	Domain-driven design as a basis for service boundaries; independent scalability and fault containment from decoupling; monitoring and testing reported by practitioners as open challenges. Event-driven coordination is treated in the wider literature rather than established by any single citation	Addresses post-agreement processing (settlement, compliance, reporting) implicitly; does not model a pre-trade, multi-party negotiation as a distinct architectural concern

Literature Thread	Representative Source(s)	Well-Established Findings	Coverage of Bond ETF Workflow Specifics
Legacy modernization (strangler fig)	Fritzsch, Bogner, Zimmermann, and Wagner (2019); Knoche and Hasselbring (2019); Ma, Li, Lee, and Lee (2022); Abgaz et al. (2023)	Empirically demonstrated incremental migration combining strangler fig with domain-driven design; bounded-context identification through domain-driven design as the organizing step; boundary identification only partially automated in general; incremental migration widely recommended; transactional concerns during modernization	General-purpose migration studies (non-financial systems); no asset-management- or ETF-specific bounded-context guidance
Bond ETF market structure	Dannhauser (2017); Finnerty, Reisel, and Zhong (2025); Koont, Ma, Pástor, and Zeng (2025)	Empirical link between creation and redemption basket inclusion and bond liquidity; mispricing arbitrage as the primary mechanism; transaction costs and inventory management as constraints; baskets of cash and a subset of index bonds, adjusted dynamically; ETF ownership associated with lower bond spreads	Describes the mechanism as a market and liquidity phenomenon; models basket choice but not negotiation, system-of-record boundaries, UI roles, or workflow representation

Note. Assessments in the final column are this review's analysis of the sources named in the second column. None of those sources is among the summary-only studies identified in Section 5.1.

4.2 Coverage Matrix Across Workflow Dimensions

Table 1 organizes the evidence by literature thread. To make the gap precise, Table 2 instead organizes it by the operational dimensions that characterize bond ETF creation and redemption, and marks whether each literature thread addresses that dimension. A dimension is marked as addressed only where a reviewed source treats it directly, as partially addressed where a source treats a close analogue, and as not addressed where no reviewed source engages it. As shown in Table 2, no single thread, and no combination of threads, covers the full set of dimensions. Three dimensions in particular, namely multi-party negotiation, shared multi-role workflow state, and dual-path settlement reconciliation, are addressed at most through close analogues, and no reviewed source treats any of them directly.

Table 2. Coverage matrix mapping each reviewed literature thread against the operational dimensions of bond ETF creation and redemption.

Workflow dimension	Micro-frontend	Microservices and EDA	Legacy modernization	Bond ETF market structure
Multi-party basket negotiation	Not addressed	Not addressed	Not addressed	Basket choice modeled; negotiation not

Workflow dimension	Micro-frontend	Microservices and EDA	Legacy modernization	Bond ETF market structure
				addressed
Shared, multi-role workflow state	Not addressed	Partial (bounded contexts)	Not addressed	Not addressed
Multi-role user interface	Partial (composition only)	Not applicable	Not applicable	Not addressed
Dual-path settlement reconciliation	Not addressed	Partial (event patterns)	Partial (transactional concerns)	Not addressed
Same-day NAV and reporting deadline	Not addressed	Partial (async processing)	Not addressed	Implied, not as a systems need
Auditability and traceability	Not addressed	Partial (observability noted)	Not addressed	Not addressed
Legacy migration path	Not applicable	Partial (general guidance)	Addressed (general, non-financial)	Not applicable

Note. Entries draw on the sources named for each thread in Table 1, together with Lando and Hasselbring (2025) for the micro-frontend thread. Every entry, including those marked Not addressed, is this review's analysis of those sources and not a finding reported by them. Li, Ma, and Lu (2020) and Li, Zhang, and colleagues (2021) were consulted only through published summaries and support no entry.

4.3 Comparative Discussion: How This Review Differs from Existing Studies

The existing micro-frontend reviews by Peltonen, Mezzalira, and Taibi (2021) and Cunha de Amorim and Canedo (2025) are themselves broad, cross-industry syntheses. They are not focused on financial services and make no claim to be. The broad microservices literature, represented here by Dragoni et al. (2017), Jamshidi et al. (2018), Waseem et al. (2021), and Abgaz et al. (2023), is general, covering microservice evolution, quality attributes, and decomposition across application domains without reference to any one industry. The migration studies by Fritzsch, Bogner, Zimmermann, and Wagner (2019), Knoche and Hasselbring (2019), and Ma, Li, Lee, and Lee (2022) are grounded in general or non-financial settings. The bond ETF studies by Dannhauser (2017), Finnerty, Reisel, and Zhong (2025), and Koont, Ma, Pástor, and Zeng (2025) are empirical finance papers with no software-architecture content.

The present review adds no new empirical data in any of these individual areas. Its distinguishing feature is the juxtaposition of all four threads, organized around a single set of review questions that target one enterprise workflow. The gap becomes visible only in that juxtaposition, which is why Tables 1 and 2 present the

same evidence along two axes. Each literature is internally coherent and individually well evidenced, yet none was written with reference to the others, and as Table 2 records, no reviewed source addresses a negotiated, multi-party, role-differentiated workflow with two settlement paths operating under same-day NAV and reporting deadlines.

4.4 Identified Gaps

Gap 1: No domain-specific architectural model for the basket negotiation workflow

The basket negotiation between the AP and the portfolio manager runs through proposal, counter-proposal, partial acceptance, and tracking-error validation, and it has no counterpart treatment in the reviewed software-engineering literature. In the accounts reviewed here, including the description of event-driven architecture by Jamshidi et al. (2018), event-driven coordination is characterized around single-event triggers and does not model a business object that is iteratively revised by two counterparties before a triggering event exists. This characterizes the examples reviewed and is not a general limitation of event-driven architecture. Whether this negotiation stage should be modeled as its own bounded context, as a stateful process spanning several existing contexts, or as an application-layer workflow engine sitting above otherwise event-driven services is not addressed by any source reviewed here.

Gap 2: No micro-frontend treatment of role-differentiated, multi-step approval interfaces

The micro-frontend literature reviewed here, including Peltonen, Mezzalira, and Taibi (2021) and Cunha de Amorim and Canedo (2025), is dominated by cases where composition is spatial or navigational, meaning independent micro-frontends occupy different pages or regions of a layout. Bond ETF workflow management instead requires several distinct roles, such as trader or AP liaison, portfolio manager, compliance officer, and operations analyst, to interact with the same evolving object, the proposed basket, at different points in its lifecycle, with different permitted actions and data visibility at each point. None of the reviewed sources addresses shared, mutable workflow state across independently deployed, independently versioned micro-frontend modules.

Gap 3: Reconciliation of dual settlement paths is unaddressed as a service-boundary problem

The duality of in-kind and cash dealing in bond ETF creation and redemption means two structurally different operational paths must resolve to a single fund position and a single NAV calculation on the same business day. The event-driven and domain-driven design patterns described in the reviewed microservices literature are plausible tools for this reconciliation problem. No source reviewed here, however, applies them to a dual-settlement-path reconciliation problem specifically. The bond ETF market-structure literature represented by Finnerty, Reisel, and Zhong (2025) does not address it as a systems question at all.

Gap 4: Legacy modernization guidance is generic rather than sequenced to asset management bounded contexts

The migration study by Ma, Li, Lee, and Lee (2022) gives general methodological guidance on combining the strangler fig pattern with domain-driven design, but it is grounded in non-financial legacy systems, and the survey by Knoche and Hasselbring (2019) examines migration drivers and barriers without reference to any financial workflow. The broader decomposition review by Abgaz et al. (2023) confirms that boundary identification is not yet a solved or fully automated problem in general. None of this work indicates what the correct bounded contexts are for an asset management platform's bond ETF workflow specifically, or in what sequence they should be extracted given the domain's same-day NAV and reporting dependencies.

Gap 5: Absence of literature jointly evaluating the micro-frontend and microservices layers together for one regulated, negotiation-heavy workflow

Even setting ETF specifics aside, the reviewed literature studies micro-frontend decomposition and backend microservices decomposition as separate research areas and does not evaluate them together as one coordinated architecture for a single workflow. This gap was identified within the sources reviewed. It compounds the ETF-specific gaps above, because a reference architecture for bond ETF workflow management would need to establish this joint pattern, and no evaluated example of it was found among the reviewed sources. Gap 5 is therefore the broadest of the five, and a study addressing it would also inform the other four.

5. Conclusion

Micro-frontend architecture and microservices with event-driven backends are each substantiated by peer-reviewed literature, including systematic mapping studies, systematic literature reviews, a survey in IEEE Software, and empirical strangler-fig migration case studies. Bond ETF creation and redemption is, separately, a well-studied market structure and liquidity phenomenon in finance journal literature. This review's central finding is that, within the sources reviewed, these bodies of work have not been examined together. The location of the gap matters. It coincides with the operational features that make bond ETF workflow management distinctive, namely negotiated, multi-party basket construction, dual settlement-path reconciliation, and role-differentiated approval under same-day NAV and reporting deadlines.

5.1 Limitations

As noted in Section 3.1, this is a narrative rather than a formally registered systematic review, so it is possible that additional relevant peer-reviewed sources were not surfaced. Restricting the evidence to peer-reviewed and indexed academic work ensures a defensible citation base, but it necessarily excludes the substantial body of practitioner knowledge on these topics that has not been published in academic venues, and it narrows the reference list accordingly. The absence of literature at a given intersection, for example the joint pattern discussed under Gap 5, is reported as a finding of the search rather than as proof that no such work exists. Relevant material may reside in venues not covered here, including proprietary vendor architecture documentation or internal technical reports that are not publicly available. Two cited studies, Li, Ma, and Lu (2020) and Li, Zhang, and colleagues (2021), are drawn on only at the level of their published summaries and are cited for context, not as evidence for any finding.

A further and more fundamental limitation concerns the paper's own premise. The gap analysis rests on specific operational properties of the bond ETF creation and redemption workflow, namely the three distinguishing properties set out in the Introduction. The finance sources cited here establish the market and liquidity behavior of that mechanism, and Koont, Ma, Pástor, and Zeng (2025) document that funds choose baskets of cash and a subset of index bonds and adjust them dynamically. Rule 6c-11 supplies part of what that literature does not, since it conditions custom basket use on review by named roles and permits a later basket to differ from the initial basket used on the same business day (Securities and Exchange Commission, 2019). The multi-round negotiation between the authorized participant and the portfolio manager is undocumented in the cited sources and is stated as an assumption defining the review's scope (Section 3.5). A stronger version of this work would ground that negotiation in a primary source dedicated to it, and its absence limits the evidentiary base for the gaps in Section 4.4.

5.2 Actionable Directions for Future Work

The following directions follow from the gaps identified in Section 4.4 and the uncovered dimensions shown in Table 2. They are offered as a starting agenda for design-oriented research, since this paper establishes that the gaps exist without proposing a solution.

- Develop and empirically evaluate a bounded-context model for the AP and portfolio-manager basket negotiation, treating it as a first-class workflow object instead of an implicit precursor to a single trade-execution event (Gap 1).
- Investigate micro-frontend state-coordination patterns suitable for role-differentiated, multi-step approval interfaces, extending existing session-propagation guidance from the general micro-frontend literature to cover shared, mutable workflow state (Gap 2).
- Examine whether event sourcing and command query responsibility segregation (CQRS) patterns, documented in the general microservices literature, can be adapted specifically to reconcile the dual cash and in-kind settlement paths against one fund position and NAV calculation (Gap 3).
- Extend strangler-fig migration case-study methodology, as demonstrated by Ma, Li, Lee, and Lee (2022) in a non-financial context and situated within the broader decomposition evidence reviewed by Abgaz et al. (2023), to an asset management platform, in order to establish domain-specific bounded-context sequencing under real regulatory constraints (Gap 4).

Design and evaluate a reference architecture that

jointly specifies both the micro-frontend and the microservices layers for one such workflow, addressing the joint-evaluation gap recorded in Section 4.4 as well as the ETF-specific gaps (Gap 5).

References

1. Abgaz, Y., McCarren, A., Elger, P., Solan, D., Lapuz, N., Bivol, M., Jackson, G., Yilmaz, M., Buckley, J., & Clarke, P. (2023). Decomposition of monolith applications into microservices architectures: A systematic review. IEEE Transactions on Software Engineering, 49(8), 4213–4242. https://doi.org/10.1109/TSE.2023.3287297

2. Ben-David, I., Franzoni, F., & Moussawi, R. (2017). Exchange-traded funds. Annual Review of Financial Economics, 9(1), 169–189. https://doi.org/10.1146/annurev-financial-110716-032538

3. Cunha de Amorim, G., & Canedo, E. D. (2025). Micro-frontend architecture in software development: A systematic mapping study. In Proceedings of the 27th International Conference on Enterprise Information Systems (Vol. 2, pp. 105–116). SciTePress. https://doi.org/10.5220/0013195800003929

4. Dannhauser, C. D. (2017). The impact of innovation: Evidence from corporate bond exchange-traded funds (ETFs). Journal of Financial Economics, 125(3), 537–560. https://doi.org/10.1016/j.jfineco.2017.06.002

5. Dragoni, N., Giallorenzo, S., Lluch Lafuente, A., Mazzara, M., Montesi, F., Mustafin, R., & Safina, L. (2017). Microservices: Yesterday, today, and tomorrow. In Present and ulterior software engineering (pp. 195–216). Springer. https://doi.org/10.1007/978-3-319-67425-4_12

6. Finnerty, J. D., Reisel, N., & Zhong, X. (2025). ETFs, creation and redemption processes, and bond liquidity. Journal of Financial and Quantitative Analysis, 60(4), 1891–1924. https://doi.org/10.1017/S0022109024000346

7. Fritzsch, J., Bogner, J., Zimmermann, A., & Wagner, S. (2019). From monolith to microservices: A classification of refactoring approaches. In Software engineering aspects of continuous development and new paradigms of software production and deployment (Lecture Notes in Computer Science, pp. 128–141). Springer. https://doi.org/10.1007/978-3-030-06019-0_10

8. Jamshidi, P., Pahl, C., Mendonça, N. C., Lewis, J., & Tilkov, S. (2018). Microservices: The journey so far and challenges ahead. IEEE Software, 35(3), 24–35. https://doi.org/10.1109/MS.2018.2141039

9. Knoche, H., & Hasselbring, W. (2019). Drivers and barriers for microservice adoption: A survey among professionals in Germany. Enterprise Modelling and Information Systems Architectures, 14(1), 1–35. https://doi.org/10.18417/emisa.14.1

10. Kojo, R. H. H., Real, L. F. C., Ferreira, R. C., de Oliveira Rosa, T., & Goldman, A. (2026). Exploring micro frontends: A case study application in e-commerce. In D. Bianculli et al. (Eds.), Software architecture: ECSA 2025 tracks and workshops (Lecture Notes in Computer Science, Vol. 15982, pp. 187–201). Springer. https://doi.org/10.1007/978-3-032-04403-7_17

11. Koont, N., Ma, Y., Pástor, Ľ., & Zeng, Y. (2025). Steering a ship in illiquid waters: Active management of passive funds. The Review of Financial Studies, 38(10), 2887–2935. https://doi.org/10.1093/rfs/hhaf034

12. Lando, B., & Hasselbring, W. (2025). Toward bundler-independent module federations: Enabling typed micro-frontend architectures (arXiv preprint arXiv:2501.18225). https://doi.org/10.48550/arXiv.2501.18225

13. Li, C.-Y., Ma, S.-P., & Lu, T.-W. (2020). Microservice migration using strangler fig pattern: A case study on the Green Button system. In 2020 International Computer Symposium (ICS) (pp. 519–524). IEEE. https://doi.org/10.1109/ICS51289.2020.00107

14. Li, S., Zhang, H., Jia, Z., Zhong, C., Zhang, C., Shan, Z., Shen, J., & Babar, M. A. (2021). Understanding and addressing quality attributes of microservices architecture: A systematic literature review. Information and Software Technology, 131, 106449. https://doi.org/10.1016/j.infsof.2020.106449

15. Ma, S.-P., Li, C.-Y., Lee, W.-T., & Lee, S.-J. (2022). Microservice migration using strangler fig pattern and domain-driven design. Journal of Information Science and Engineering, 38(6), 1285–1303. https://doi.org/10.6688/JISE.202211_38(6).0010

16. Peltonen, S., Mezzalira, L., & Taibi, D. (2021). Motivations, benefits, and issues for adopting micro-frontends: A multivocal literature review. Information and Software Technology, 136, 106571. https://doi.org/10.1016/j.infsof.2021.106571

17. Securities and Exchange Commission. (2019). Exchange-traded funds (Release Nos. 33-10695, IC-33646; File No. S7-15-18). 84 Fed. Reg. 57162, codified at 17 C.F.R. § 270.6c-11. https://www.federalregister.gov/d/2019-21250

18. Taibi, D., & Mezzalira, L. (2022). Micro-frontends: Principles, implementations, and pitfalls. ACM SIGSOFT Software Engineering Notes, 47(4), 25–29. https://doi.org/10.1145/3561846.3561853

19. Waseem, M., Liang, P., Shahin, M., Di Salle, A., & Márquez, G. (2021). Design, monitoring, and testing of microservices systems: The practitioners' perspective. Journal of Systems and Software, 182, Article 111061. https://doi.org/10.1016/j.jss.2021.111061