Designing a Conceptual Integration Architecture for Bond Transactions Using DDD and EDA Based on Operational Integration Evidence: A Case Study at an Indonesian Bank

Adam Wahidin

Information Systems Management, BINUS Graduate Program, Bina Nusantara University, Jakarta, DKI Jakarta, Indonesia
E-mail: adam.wahidin@binus.ac.id

(Received: 19 Aug 2026, revised: 25 Aug 2026, 29 Aug 2026, 3 Sep 2026, accepted: 8 Sep 2026)

Abstract

Bond transaction processing in banking depends on coordinated exchanges of instructions, reference data, processing outcomes, and status across several applications. This study develops a conceptual integration architecture and traces major design decisions to operational integration evidence. A descriptive qualitative case study at an anonymized Indonesian bank analyzed 275 records from January to December 2025: 225 system logs and 50 ticket/email complaints, supported where available by root-cause analyses and process/architecture documents. The corpus comprised 135 inter-system timeouts, 59 data/status synchronization issues, 50 pending processes, and 31 manual reconciliations. These categories were treated as operational signals rather than technical root causes. Contextual interpretation yielded four bounded integration indications and six integration requirements. Strategic Domain-Driven Design defined six bounded contexts and explicit responsibilities; conceptual Event-Driven Architecture modeled business facts and status semantics while retaining request/response where an immediate result was required. The resulting conceptual architecture separates response-dependent interaction from selective event/status propagation. A seven-session Focus Group Discussion involving 20 practitioners from the case context evaluated the resulting requirements, DDD/EDA artifacts, and conceptual architecture against seven predefined conceptual-evaluation aspects. Because the participants remained part of the case context, the FGD is treated as conceptual evaluation rather than independent technical validation. The scientific contribution is an integrative evidence-to-design reasoning sequence combining bounded evidence interpretation, responsibility-before-events sequencing, and forward/backward traceability. Implementation readiness and non-functional characteristics were not evaluated.

Keywords: System Integration Architecture, Domain-Driven Design, Event-Driven Architecture, Bond Transaction, Traceability.

I. INTRODUCTION

Information systems research conceptualizes the information systems artifact through a general-systems perspective [1], while enterprise systems support cross-functional organizational processes [2]. In transaction-intensive settings, integration must preserve the meaning and timing of instructions, reference data, outcomes, and statuses required for process continuity. Because business processes evolve with organizational and technical conditions [3], integration problems are interpreted here in the context of the process that produces them.

Bond transactions involve instruction intake, reference-data validation, transaction processing, treasury coordination, financial recording, settlement-related status updates, and reconciliation. Settlement also depends on coordinated securities-delivery and payment information [4]. Across heterogeneous applications, integration must therefore preserve information exchange and process continuity, consistent with persistent software-unit and business-process integration challenges [5], [6].

The analyzed records contain four recurring operational patterns: inter-system timeout, data/status synchronization issues, pending processes, and manual reconciliation. These are evidence, not architectural diagnoses. Logs reveal runtime behavior [7], data-integration quality is context-dependent [8], and event logs must be understood relative to process structures [9]. Accordingly, records are interpreted in process

DOI: 10.34148/teknika.v15i3.1549
TEKNIKA, Volume 15(3), November 2026, pp. 449-459
ISSN 2549-8037, EISSN 2549-8045

CC BY SA

and integration context before bounded indications and requirements are derived.

DDD structures domain language, responsibilities, and bounded contexts [10], [11], while EDA models meaningful changes and their dissemination. Because event-driven designs involve modularity and performance trade-offs [12], [13], this study does not prescribe asynchronous communication. Integration needs are derived first; DDD then structures responsibility boundaries, after which EDA is applied selectively to relevant business facts and status visibility.

Prior studies separately address operational evidence interpretation [7], [8], [9], DDD boundary design [10], [11], and EDA trade-offs [12], [13]. A combined DDD-EDA reference architecture has also been proposed [14], while conformance research examines downstream consistency between domain-model and API decisions [15]. These streams do not make the upstream case-specific evidence-to-design chain their primary focus. The main contribution is a case-grounded, staged, and traceable evidence-to-design process that derives architecture-neutral requirements from contextually interpreted operational integration evidence, then uses DDD to structure responsibility boundaries before selectively applying EDA to interaction semantics.

The study is organized around two research questions. RQ1 asks which bounded integration indications can be identified in the Bank XYZ bond-transaction process from process information, operational records, integration points, and supporting evidence. RQ2 asks what conceptual integration architecture can be constructed with DDD and EDA from the resulting requirements. For RQ2, DDD and EDA were used as complementary conceptual lenses. They were not inferred from the records or treated as inherently superior to service-oriented architecture (SOA), enterprise service bus (ESB) integration, business-process redesign, or other approaches. The architecture was not predetermined. Evidence was interpreted first into bounded integration indications and architecture-neutral process/information requirements, after which DDD structured responsibility and EDA was applied selectively. Across that sequence, three controls are retained: bounded interpretation before requirement derivation, responsibility before event modeling, and forward/backward traceability grounded in the 275-record corpus.

II. RESEARCH METHODOLOGY

A. Research Design and Case Context

A descriptive qualitative case design using document analysis was used because RQ1 and RQ2 require contextual interpretation of operational evidence and system interactions rather than statistical estimation. The case is Bank XYZ's bond-transaction process and related system integrations; the evidence window covers January to December 2025. Document analysis used rule-based qualitative categorization followed by contextual interpretation under controlled access, confidentiality, and anonymization. The unit of business analysis is the bond-transaction process; the unit of evidence is a retained analytical record assigned one primary operational category. Additional symptoms remain contextual rather than separate findings. Counts describe the corpus and do not determine architecture requirements; no inferential testing was used.

B. Data Sources, Selection, and Integrity Controls

The final corpus contains 275 records from January to December 2025: 225 system logs and 50 ticket/email complaints. Process and architecture documents located activities, applications, and integration points. Available RCA documents served only as supporting evidence for linked records or contexts and were not generalized to the corpus. These sources therefore served distinct evidential functions.

The complete annual operational window available within the approved case-study scope was used. Records in the defined source set were screened in a fixed analytical order; all eligible records were retained after screening and deduplication, with no further sampling. A record had to fall within the study period, relate directly to bond transactions, and contain minimum usable context. The required context comprised a locatable occurrence, an identifiable system/integration context, and documented operational or follow-up evidence relevant to integration, data/status exchange, process continuation, or reconciliation. Out-of-scope, non-integration, and context-deficient records were excluded.

Potential duplicates were compared using anonymized process/transaction references, time, integration point, symptom description, and follow-up evidence. Records were merged only when the evidence supported the same operational occurrence. Each retained record received one primary category based on the directly documented operational trigger; additional symptoms remained contextual. For example, a timeout followed by a pending state remained classified as timeout when timeout was the documented trigger. Likewise, reconciliation performed after a documented data mismatch did not replace mismatch as the primary category. Ambiguous records were interpreted conservatively using process context and supporting evidence. Provenance and selection/deduplication decisions are retained in the controlled register. The 275 records therefore constitute the complete post-screening analytical corpus obtained from the defined 2025 source set after applying the stated eligibility and deduplication criteria. The number 275 was not a predetermined sample size and is not used as a denominator for incident prevalence.

The retained records were classified within their relevant process and system areas by staff-level practitioners familiar with those operational contexts, using a common set of classification rules. The researcher then consolidated the area-level classifications against the documentary evidence to maintain consistency across the corpus. When a record involved multiple system areas or its primary operational trigger was unclear, the relevant staff-level system perspectives were considered during consolidation. Differing interpretations were resolved by re-examining the documentary evidence, process sequence, integration context, and the same classification rules. This contextual cross-

CC BY SA

checking did not constitute independent inter-rater recoding: the same records were not independently classified in parallel for agreement assessment, and no inter-rater agreement statistic is reported or claimed.

For publication, personal, account, transaction, user, communication, credential, and sensitive configuration identifiers are not disclosed; system areas use anonymized labels. The controlled analytical register retains source provenance, finding IDs, primary categories, selection/deduplication decisions, and restricted evidence locations. It was progressively extended from process/integration context and IA links to REQ, responsibility/BC, and BE/interaction links. At manuscript level, reported IA, REQ, DDD/EDA, and architecture mappings provide publication-level traceability while record-level provenance remains controlled. TI denotes integration point; KR reconciliation context; IA bounded integration indication; REQ integration requirement; BC bounded context; and BE business event.

Supplementary Material S1 documents the classification and ambiguity-handling rules, evidence-to-indication criteria, requirement-derivation and consolidation criteria, forward/backward traceability structure, and a worked example. It supports inspection and reapplication of the non-sensitive analytical procedure without disclosing source records. S1 also specifies the steps needed to reproduce the analysis when authorized access to the relevant underlying evidence is available. It does not introduce a new method or evidence. The evidence-preparation and integrity controls are summarized in Table 1.

Table 1. Evidence Preparation and Integrity Controls

Control	Implementation in this Study
Inclusion/ Exclusion	Checked period, bond-process relevance, integration context, and minimum evidence; excluded out-of-scope, non-integration, and context-deficient records. Potential duplicates followed the procedure below.
Deduplication	Compared anonymized transaction/process reference, time, integration point, symptom, and follow-up evidence before merging.
Classification	Staff-level system-area practitioners applied common rules; the researcher consolidated against documentary evidence; one primary category was retained and secondary symptoms remained contextual.
Ambiguity / Disagreement	Cross-system or unclear-trigger cases used relevant system perspectives; differing interpretations returned to documentary evidence, process sequence, integration context, and common rules.
Confidentiality	Suppressed personal, account, transaction, credential, and sensitive configuration identifiers; used anonymized system-area labels.
Traceability	The controlled register retained provenance, selection/deduplication decisions, finding IDs, categories, and restricted evidence locations; it progressively linked evidence/context to IA, IA/context to REQ-02-REQ-06, the cross-cutting rationale to REQ-01, REQ to responsibility/BC, and BC/REQ to BE/interaction. The final backward check followed architecture synthesis to assess trace continuity.

C. Evidence-to-Requirement Analysis

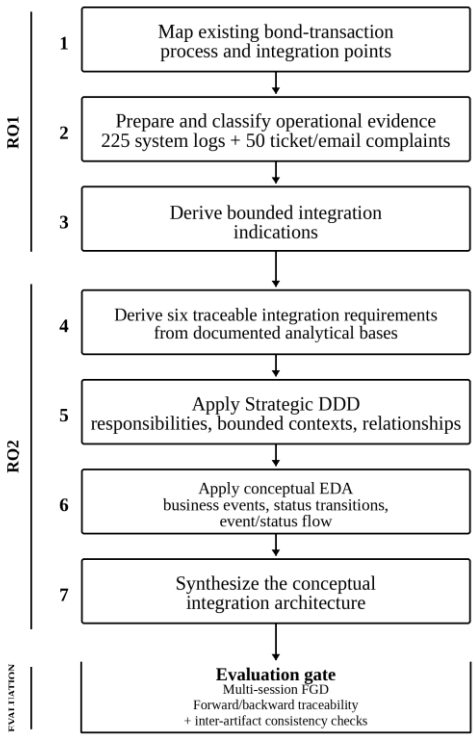

Figure 1. Evidence-to-architecture Research and Evaluation Workflow

Figure 1 summarizes seven analysis-and-design stages plus a conceptual evaluation gate. Stages 1-3 address RQ1 by mapping process/integration points, preparing evidence, and deriving bounded indications before architecture decisions. Stages 4-7 address RQ2 by deriving six requirements, establishing Strategic DDD responsibilities, modeling cross-context facts with conceptual EDA, and synthesizing the architecture. Trace links were recorded progressively. At the evaluation gate, the FGD evaluated the conceptual artifacts, while forward/backward checks assessed trace continuity and

DOI: 10.34148/teknika.v15i3.1549
TEKNIKA, Volume 15(3), November 2026, pp. 449-459
ISSN 2549-8037, EISSN 2549-8045

the documented upstream basis of each reported architecture element.

During bounded-indication derivation, the analytical question was which bounded integration indication the evidence could support, not which technology should be introduced. Four rules constrained interpretation. Timeout evidence supported IA-01 only when continuation depended on an immediate inter-system response; it indicated an interaction-level direct-response dependency, not a technical cause. Pending evidence supported IA-02 when continuation was blocked by an unavailable status, response, or prerequisite; this process-level waiting was not treated as timeout. Data/status differences indicated misalignment only where contexts should be comparable. Manual reconciliation could indicate limited alignment or status propagation without implying elimination of manual work.

REQ-02 to REQ-06 were retained only when grounded in a bounded integration indication and context. They were stated as process/information needs rather than implementation mechanisms, and unobserved root-cause claims were excluded. REQ-01 followed a separate cross-cutting responsibility rationale. A timeout at TI-01 or TI-04, for example, supported IA-01 only when continuation depended on a direct response. This interpretation was translated into REQ-02 without prescribing events, asynchronous transport, or a technical cause. Likewise, a data/status difference at TI-02 or TI-03 supported IA-03 only where the contexts should align. This interpretation was translated into REQ-03/REQ-05 without asserting an authoritative system or mismatch cause. Requirements were expressed in architecture-neutral process/information terms before DDD/EDA constructs and consolidated according to process/information need rather than category frequency.

Traceability evolved with the same sequence. Stages 1-2 linked findings to process/TI-KR contexts, provenance, and categories. Stage 3 added evidence/context-to-IA links, and Stage 4 added IA/context-to-REQ links, with REQ-01 recorded separately. Stage 5 linked requirements to responsibilities and bounded contexts. Stage 6 added producer/interested-context, status-transition, event, and interaction links. Stage 7 linked retained DDD/EDA artifacts to architecture relationships. The final backward check examined these prior links rather than generating rationale retrospectively.

Independent parallel recoding was not performed, so subjectivity was constrained procedurally rather than assessed through inter-rater agreement. Classification used common rules, relevant system-area perspectives, documentary re-examination, process sequence, and integration context. For REQ-02 to REQ-06, contextual interpretation required a bounded indication/context and architecture-neutral formulation; REQ-01 followed the separate cross-cutting rationale, while frequency-only and unsupported causal reasoning were excluded. These controls constrained rather than eliminated interpretive discretion.

D. DDD, EDA, and Architecture Synthesis

Stage 4 was completed before the DDD/EDA frameworks were applied. Requirements were first stated in process/information terms. Bounded-context and event/status constructs were introduced only afterward. Strategic DDD then used six analytical steps: domain scope, ubiquitous language, responsibility grouping, bounded-context definition, context relationships, and data/status responsibility. These criteria were applied to derived requirements and responsibility candidates. Candidates serving one coherent purpose were grouped; materially different responsibilities or data/status meanings remained separate. Existing applications served only as current-state anchors. The resulting groups were checked for requirement coverage and needed relationships, keeping bounded-context discovery tied to derived needs rather than application boundaries alone [10], [11].

Only after those responsibility boundaries were established did conceptual EDA answer which business facts or status changes had to become visible across contexts. In this study, conceptual EDA was operationalized through six analytical steps: business-event identification, status transition, producer context, interested context, referenced data/status, and event/status flow. A business event is a fact that has occurred; a command requests another context to act; request/response is the communication mode used when an immediate answer is required. Thus, event semantics do not by themselves imply asynchronous transport. Direct interaction is retained for response-dependent steps, while event/status propagation is reserved for the relationship between Transaction Processing and Status and Reconciliation. Broker, topic, queue, payload, retry, delivery guarantee, throughput, and deployment choices remain outside the conceptual scope.

E. FGD Conceptual Evaluation

The conceptual FGD involved 20 practitioners purposively selected to provide complementary perspectives across the case. Eighteen participants covered six anonymized system areas; each area was represented by one internal staff-level practitioner, one internal unit-head-level practitioner, and one vendor-side Solution Architect or IT consultant supporting that system. Two additional internal participants, one staff-level and one unit-head-level, represented system integration. Internal staff-level participants had at least five years of relevant experience, and unit-head-level participants at least ten years. Staff-level practitioners who had contributed to the system-area evidence review later supplied operational and system-context observations. Unit heads contributed supervisory evaluation, while vendor specialists contributed system-specific technical perspectives. The researcher prepared the review package, facilitated comparison with the evaluation aspects and conceptual artifacts, and consolidated observations and dispositions. These different roles broadened the evaluation perspective, but they did not provide evaluator independence because all evaluators remained connected to the case or the systems assessed.

The FGD was conducted only after the evidence analysis, integration requirements, DDD/EDA artifacts, and draft conceptual architecture were complete. It comprised seven

sessions: six system-area review sessions corresponding to the six anonymized system areas and a seventh cross-system review session involving all 20 participants. Each session lasted approximately 60 to 120 minutes depending on the review scope. Across the seven sessions, the evaluation process comprised review-package preparation, system-area review, collection of practitioner observations, cross-system discussion, consolidation, and a final traceability/inter-artifact consistency check. The review package covered the current integration context, 275-record profile, available RCA, indication/requirement mappings, DDD/EDA artifacts, conceptual architecture, and traceability matrix.

Before the formal FGD sessions, seven qualitative aspects were defined from the conceptual-evaluation objective and the evidence-to-design artifacts: context relevance, evidence-to-design alignment, domain boundaries, data/status responsibility, interaction semantics, cross-system consistency, and traceability/conceptual plausibility. The researcher prepared the evaluation frame, which unit-head-level practitioners and vendor-side system specialists reviewed for clarity and case relevance. Observations were organized by these aspects and consolidated qualitatively. Where perspectives differed, they were compared with the corresponding conceptual artifacts; conceptual issues were clarified, whereas implementation-dependent matters were deferred. The study did not use a separate coding scheme, voting, numerical scoring, or forced consensus. Because the design already existed before the FGD, the sessions evaluated the requirements, DDD/EDA artifacts, and architecture rather than generating them. The resulting evidence is therefore case-based conceptual evaluation across the represented roles and system areas, not statistical representativeness or independent technical validation.

III. RESULTS AND DISCUSSION

A. Bounded Integration Indications

Table 2 summarizes the corpus: 49.09% inter-system timeouts, 21.45% data/status synchronization issues, 18.18% pending processes, and 11.27% manual reconciliations. These percentages describe the analyzed corpus, not incident probabilities. Records were then located in anonymized process/integration contexts spanning Channel/Front End, Wealth System, Customer Information, Core Financial, Treasury System, Trading Platform, and cross-process reconciliation. Figure 2 shows the current-state context used before bounded indications were derived.

TI-01 is Channel/Front End-to-Wealth instruction transfer; TI-02 reference validation; TI-03 financial recording; TI-04 Wealth-Treasury coordination; and TI-05 Treasury-Trading Platform interaction. KR-01 is a cross-process reconciliation context, not an application; the labels preserve analytical location without exposing endpoints or configuration details.

Contextual interpretation produced four bounded indications: IA-01 direct-response dependency; IA-02 waiting for unavailable status, response, or prerequisite; IA-03 data/status misalignment in comparable contexts; and IA-04 limited status propagation requiring cross-process traceability. IA-01 and IA-02 remain distinct, and neither substitutes for technical root-cause evidence. These indications answered RQ1 and entered Stage 4. Table 3 shows their many-to-many mappings to REQ-02 to REQ-06 and the separate rationale for REQ-01; six requirements resulted from formulation and consolidation.

Table 2. Distribution of the Final Operational Evidence Corpus

Primary category	n	%	Analytical role
Inter-system timeout	135	49.09	Signal for direct-response dependency analysis
Data/status synchronization issue	59	21.45	Signal for data/status alignment analysis
Pending process	50	18.18	Signal for process-level waiting / unmet prerequisite analysis
Manual reconciliation	31	11.27	Signal for cross-system data/status alignment and traceability analysis

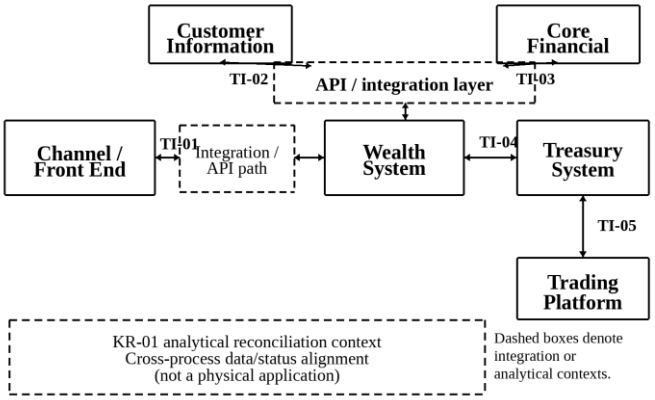

Figure 2. Anonymized Current-State Integration Context Used For Analysis

Table 3. Condensed Analytical-basis-to-requirement Mapping

Analytical Basis	Integration Context	Interpretation / IA / Limit	Req.
Inter-system timeout	TI-01; TI-04	IA-01 direct-response dependency; no specific technical cause inferred.	REQ-02
Data/status sync issue	TI-02; TI-03	IA-03 misalignment in comparable contexts; technical cause/source of truth not established.	REQ-03; REQ-05

DOI: 10.34148/teknika.v15i3.1549

CC BY SA

Analytical Basis	Integration Context	Interpretation / IA / Limit	Req.
Pending process	TI-04 / waiting point	IA-02 waiting for status/response/prerequisite; not automatically timeout; no performance cause established.	REQ-02; REQ-04
Manual reconciliation	KR-01 cross-process reconciliation context	IA-04 limited status propagation/traceability; no implication to eliminate manual work.	REQ-04; REQ-06
Cross-cutting responsibility rationale	TI-01 to TI-05; KR-01	Process-responsibility boundaries across mapped contexts; separate from IA-01 to IA-04.	REQ-01

Table 3 condenses the analytical-basis-to-requirement mapping. REQ-02 to REQ-06 have many-to-many indication/context mappings, whereas REQ-01 follows the separate cross-cutting responsibility rationale. A publication-safe TI-04 example shows Transaction Processing awaiting a Treasury-coordination result; this supported IA-01 and REQ-02 without inferring an infrastructure, capacity, network, or application cause. The interaction mechanism was selected only later during DDD/EDA synthesis. Candidate requirements were retained only when grounded in a bounded indication/context, expressed as architecture-neutral process/information needs, and free of unsupported causal or implementation claims. Equivalent needs were consolidated, yielding six non-duplicate requirements: REQ-01 responsibility boundaries; REQ-02 coordination flow; REQ-03 data/status and validation responsibility; REQ-04 traceable status changes; REQ-05 coordinated information/data-status; and REQ-06 reconciliation/revalidation traceability. The count was an analytical outcome, not a predefined design target. DDD/EDA were applied only after the requirements and upstream bases were established.

B. Strategic DDD Results

DDD produced six responsibility groups and six conceptual bounded contexts. Their ubiquitous language was anchored to bond-process terms: transaction instruction, customer/reference validation, transaction processing, treasury coordination, financial recording/posting, completion/settlement status, mismatch/exception, and reconciliation. Applications remained current-process anchors rather than one-to-one boundary rules. BC-05 groups the distinct Treasury System and Trading Platform anchors because both serve one treasury-coordination business capability: coordinating order/position information and the resulting treasury status returned to Transaction Processing. Its shared ubiquitous language centers on treasury coordination, order/position, and treasury status; responsibility ownership for that capability is therefore placed within BC-05 across the two anchors rather than treating each application as a separate bounded context. BC-06 retains completion/settlement status, mismatch/exception, alignment, reconciliation, and closure as one responsibility for cross-process lifecycle status alignment and reconciliation traceability. It remains distinct from transaction processing and excludes generic logging, infrastructure monitoring, observability, and technical error handling. Settlement remains completion information/status rather than a separate bounded context because the study does not redesign the operational settlement function. Table 4 summarizes these conceptual bounded contexts and their responsibilities.

Table 4. Conceptual Bounded Contexts and Responsibilities

Bounded Context / Responsibility	Existing Anchor / Principal Relationship	Req.
BC-01 Channel and Instruction - receive/record instruction	Channel/Front End; relationship with BC-02	REQ-01; REQ-02
BC-02 Transaction Processing - processing and main status	Wealth System; with BC-01, BC-03, BC-04, BC-05, BC-06	REQ-01; REQ-02; REQ-04; REQ-05
BC-03 Customer and Reference Data - reference validation	Customer Information; relationship with BC-02	REQ-01; REQ-03; REQ-05
BC-04 Financial Recording - recording/posting status	Core Financial; relationship with BC-02	REQ-01; REQ-03; REQ-05; REQ-06
BC-05 Treasury Coordination - order/position and treasury status	Treasury/Trading area; relationship with BC-02	REQ-01; REQ-02; REQ-05
BC-06 Status and Reconciliation - status, exceptions, alignment, reconciliation	Cross-process status/reconciliation; relationship with BC-02	REQ-01; REQ-04; REQ-05; REQ-06

Five principal relationships connect BC-01 to BC-02 and BC-02 with BC-03, BC-04, BC-05, and BC-06. Table 5 identifies the context maintaining each conceptual data/status representation and the contexts needing it. Where one context produces a fact and another retains it for traceability, those responsibilities remain distinct in the EDA model.

CC BY SA

Table 5. Condensed Conceptual Data/Status Responsibility Matrix

Data / Status	Responsible Context	Needed By / Integration Role
Instruction / reception status	BC-01 Channel and Instruction	BC-02: starts processing; preserves reception traceability.
Customer/reference data + validation result	BC-03 Customer and Reference Data	BC-02: uses supporting data and validation outcome.
Transaction + main processing status	BC-02 Transaction Processing	BC-04/BC-05/BC-06: coordination, recording, and traceability.
Order/position + treasury status	BC-05 Treasury Coordination	BC-02: receives treasury coordination result/status.
Financial recording/posting status	BC-04 Financial Recording	BC-02: receives recording status; supports alignment.
Completion/exception/reconciliation status retained for traceability	BC-06 Status and Reconciliation	BC-02: closure/revalidation; BE-09 is produced by BC-02 and retained by BC-06.

C. Event/Status Modeling and Conceptual Architecture

After DDD established responsibility boundaries, conceptual EDA modeled business facts and status changes across them. Nine BE-labeled semantics are defined: Transaction Instruction Received (BE-01), Reference Data Validated (BE-02), Transaction Processed (BE-03), Treasury Coordination Confirmed (BE-04), Financial Recording Completed (BE-05), Transaction Status Updated (BE-06), Data/Status Mismatch Detected (BE-07), Reconciliation Completed (BE-08), and Transaction Completed (BE-09). These labels are conceptual facts, not nine asynchronous links, and producer/interested contexts follow DDD responsibilities. BE-03 marks main-processing completion; BE-09 marks end-to-end completion after required downstream conditions, produced by BC-02 and retained by BC-06 as completion/settlement status. Technical correlation remains a detailed-design concern.

Table 6 separates business-event semantics from interaction treatment. BE-02, BE-04, and BE-05 are result facts after response-dependent interactions and are not propagated asynchronous links; BE-01 is the process-start fact. Explicit event/status propagation is limited to BC-02/BC-06 through BE-03, BE-06, BE-07, BE-08, and BE-09.

Table 6. Condensed Business-event, Context, and Status-Transition Mapping

Event	Producer -> Interested Context	Status Transition / Interaction
BE-01 Transaction Instruction Received	BC-01 -> BC-02	Not recorded -> received; process-start fact.
BE-02 Reference Data Validated	BC-03 -> BC-02	Validation -> data ready; result after request/response.
BE-03 Transaction Processed	BC-02 -> BC-06	Data ready -> processing status; propagated fact.
BE-04 Treasury Coordination Confirmed	BC-05 -> BC-02	Coordination -> confirmed; result after request/response.
BE-05 Financial Recording Completed	BC-04 -> BC-02	Recording -> available; result after request/response.
BE-06 Transaction Status Updated	BC-02 -> BC-06	Previous -> latest status; propagated fact.
BE-07 Data/Status Mismatch Detected	BC-06 -> BC-02	Comparison -> reconciliation required; propagated mismatch; no technical cause assigned.
BE-08 Reconciliation Completed	BC-06 -> BC-02	Reconciliation -> aligned; propagated result.
BE-09 Transaction Completed	BC-02 -> BC-06	Conditions fulfilled -> completed; propagated status fact.

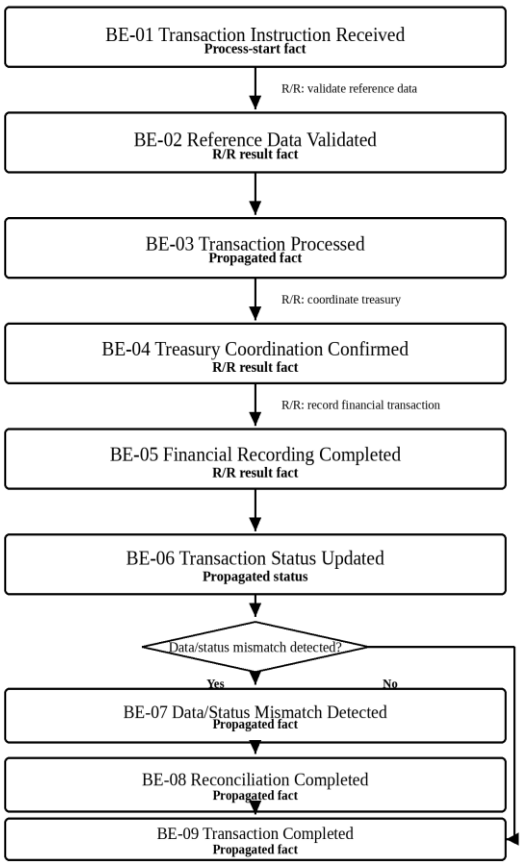

Figure 3. Conceptual Event/Status Flow for the Bond-Transaction Process

Figure 3 summarizes the conceptual sequence of business facts and status changes; R/R denotes request/response. It

visually distinguishes response-dependent result facts from theelective BC-02/BC-06 propagation path. Thus, nine BE labels do not imply nine asynchronous interactions.

Figure 4 combines bounded-context responsibility with interaction modes. Solid arrows show principal instruction or returned-result direction, not every request/response message. Customer and Reference Data, Financial Recording, and Treasury Coordination remain direct/response-dependent with Transaction Processing where continuation requires an answer; their result facts do not imply asynchronous transport. Event/status propagation is limited to BC-02/BC-06 for traceability, mismatch handling, reconciliation, and completion, with no direct propagation from Customer, Financial, or Treasury contexts to BC-06.

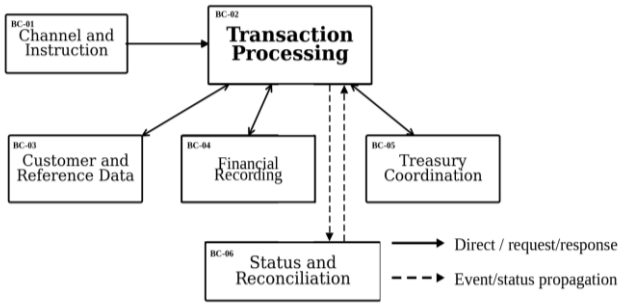

Figure 4. Proposed Conceptual Integration Architecture and Interaction Modes

The BC-02/BC-06 relationship is bidirectional conceptually, but each propagated fact remains directional: BE-03, BE-06, and BE-09 flow to BC-06, while BE-07 and BE-08 return mismatch/reconciliation outcomes. This is not an automatic closed event loop and does not claim a performance advantage.

D. Requirement-to-Architecture Traceability

Table 7 closes the reported reasoning chain by linking each requirement to its DDD boundary/relationship and EDA/interaction artifact. The publication matrix consolidates links recorded progressively during analysis rather than creating them retrospectively.

After synthesis, the reverse check traced each architecture element through its DDD/EDA basis and requirement to either a bounded indication plus operational context or, for REQ-01, the cross-cutting responsibility rationale. Tables 3-7 expose these publication-level links; record-level provenance remains controlled because banking evidence is confidential. Forward/backward traceability does not prove completeness or optimality; the backward check detects unsupported elements and explains retained ones rather than generating design rationale.

Table 7. Condensed Requirement-to-architecture Traceability

Req.	DDD Basis	EDA / Interaction Basis	Design Consequence
REQ-01	BC-01 to BC-06; explicit boundaries	Events bound to established contexts	Six responsibility contexts
REQ-02	BC-01-BC-02; BC-02-BC-05	Direct instruction; request/response for immediate result; BE-01/BE-04	Instruction and treasury paths
REQ-03	BC-03/BC-04; Table 5	Validation/recording request/response; BE-02/BE-05	Data/status responsibility; validation path
REQ-04	BC-02/BC-06 status-traceability	BE-03/BE-06/BE-09 + transitions	Status propagation to reconciliation
REQ-05	BC-02 with BC-03/04/05/06	Direct interaction vs event/status propagation	Explicit cross-context semantics
REQ-06	BC-04/BC-06; BC-06-BC-02	BE-07 mismatch; BE-08 reconciliation	Revalidation/alignment/reconciliation closure

E. FGD Evaluation Results

Across seven sessions, practitioners differed on completion-status responsibility, response-dependent interaction versus event/status visibility, and sender-receiver contract detail. Table 8 summarizes observations and dispositions. Differences were clarified conceptually; implementation-dependent matters were deferred. FGD-04 distinguished BC-02 production of BE-09 after downstream completion conditions from BC-06 retention of completion/settlement status for alignment, reconciliation, and traceability, refining semantics without structural redesign. Immediate response and cross-context status visibility were retained as complementary, while correlation/interface-contract detail was deferred. No requirement or bounded context was added or removed.

Table 8. Consolidated FGD Evaluation Across System-Area and Cross-System Sessions

Evaluation Point	Review Observation, Interpretation, And Disposition
FGD-01 Context relevance	Flow, integration points, and four categories were considered relevant; RCA remained supporting evidence. Context retained; no new requirement.
FGD-02 Evidence-to-design alignment	Participants traced IA-01 to IA-04 and REQ-01 to REQ-06 into the design; the chain was understandable across represented perspectives. Chain and requirements retained.
FGD-03 Domain boundaries	Six bounded contexts were considered reasonable strategic responsibility boundaries, not one-to-one application mappings; structure retained.

CC BY SA

Evaluation Point	Review Observation, Interpretation, And Disposition
FGD-04 Data/status responsibility	BC-02 produces the completion fact; BC-06 retains completion/settlement status for traceability. Responsibilities were distinct, not conflicting; distinction clarified and structure retained.
FGD-05 Interaction semantics	Immediate response supports continuation; resulting facts support traceability. Request/response retained for immediate dependency; event/status propagation for cross-context visibility.
FGD-06 Cross-system consistency	Sender-receiver relationships were conceptually coherent; correlation/interface-contract attributes need detail. Relationships retained; details deferred to technical design.
FGD-07 Traceability / plausibility	Architecture, relationships, data/status responsibility, event/status flow, and traceability formed a coherent case-context chain; traceability retained as case-based conceptual evaluation evidence.

F. Discussion and Contribution

The four operational patterns were treated as operational signals for contextual interpretation, not as direct evidence of technical root causes. Timeout records directed attention to response dependency, pending records to process waiting, synchronization issues to data/status alignment, and reconciliation records to status propagation. This interpretation is consistent with log-based monitoring and data-integration research emphasizing the importance of system and comparison context [7], [8].

DDD provides the domain language, responsibility grouping, bounded contexts, and relationships used to structure domain-oriented decomposition [10], [11]. In this case, responsibility boundaries, including data/status responsibilities, were established before cross-context business facts were modeled. Event identification was then aligned with that responsibility structure rather than API names, message names, or one-to-one application mappings. Existing applications therefore remain current-process anchors, while the resulting boundaries and event semantics reflect responsibility and domain meaning.

The proposed architecture is deliberately hybrid. Request/response is retained where process continuation requires an immediate result, while selective event/status propagation is used where continuing status visibility is needed. This treatment follows the derived requirements and responsibility boundaries rather than a uniform event-driven pattern. SOA/ESB-based integration and business-process redesign were not implemented, evaluated, or benchmarked in this case; the proposed design therefore does not claim to replace or outperform those alternatives. The selective use of EDA is consistent with the trade-offs reported in prior EDA studies [12], [13].

Layered traceability preserves provenance and continuity from the documented analytical basis and requirements through DDD/EDA decisions to conceptual architecture. Relative to downstream conformance checking [15], it emphasizes the upstream basis of design decisions.

Table 9 provides a structured comparison of the present study with representative DDD, EDA, combined DDD-EDA, and conformance-oriented studies using two common comparison dimensions: primary analytical focus and relationship to the present study. At the reasoning-process level, the staged sequence makes the intermediate basis of reported design decisions inspectable and separates evidence interpretation from subsequent architectural judgment.

Table 9. Structured Comparative Positioning Against Cited Approaches

Literature Stream	Primary Analytical Focus	Boundary / Relationship To Present Study
[10], [11] DDD	Domain modeling, bounded contexts, decomposition, and adoption.	Responsibility/boundary modeling; no primary upstream trace from case evidence through architecture-neutral requirements.
[12], [13] EDA	Modularity/performance trade-offs in implemented architecture styles.	Trade-off evidence; focuses on architecture qualities, not case-specific derivation from evidence, requirements, and responsibility boundaries.
[14] Combined DDD-EDA	Domain-driven, event-oriented reference architecture with technical evaluation.	Demonstrates DDD-EDA combination; the present study starts from case evidence and architecture-neutral requirements.
[15] Conformance	Automated conformance between domain-model and API endpoint design.	Downstream consistency; the present study retains upstream evidence/requirement provenance.
Present study	Case-grounded conceptual evidence-to-design reasoning.	Evidence/context -> indication or cross-cutting rationale -> architecture-neutral requirement -> DDD responsibility -> business-fact/interaction semantics -> conceptual architecture; staged traceability preserves provenance.

For banking system architects, REQ-01 to REQ-06 provide conceptual design checks, while the six bounded contexts define responsibility boundaries, principal relationships, and data/status responsibilities that inform the later definition of interface contracts. The hybrid interaction model shows where response-dependent interaction is retained and where selective event/status propagation is used for cross-context visibility. Together, these outputs provide inputs for specifying

DOI: 10.34148/teknika.v15i3.1549
TEKNIKA, Volume 15(3), November 2026, pp. 449-459
ISSN 2549-8037, EISSN 2549-8045

event/status contracts and correlation mechanisms in subsequent technical design.

Operationalization requires reliability, latency, availability, consistency, security, auditability, idempotency, failure recovery, and scalability evaluation. Without it, service levels, failure/duplicate behavior, runtime consistency, security and audit controls, and load capacity remain unestablished; design traceability is not runtime auditability.

G. Validity Boundaries and Transferability

Vendor-side Solution Architects and IT consultants added system-specific technical perspectives under the contractual and confidentiality arrangements associated with their support roles. They were external to the bank organization but remained directly involved with the systems being assessed, so their participation was not treated as independent validation. No architecture/domain expert without such involvement had equivalent authorization to access the confidential case evidence during the study. The FGD supports contextual plausibility, conceptual coherence, cross-system consistency, and traceability, but not independent corroboration of the architecture or reasoning chain. Potentially shared organizational or system assumptions within the case were not externally tested. Independent review would require authorized access under appropriate confidentiality controls.

These findings should be read within the limits of one banking organization, one bond-transaction context, the January to December 2025 evidence period, and the supporting material available in that case. Categorization relied on common rules, conservative interpretation, contextual cross-checking, and researcher consolidation; the records were not independently recoded in parallel and no inter-rater agreement assessment was performed. Accordingly, residual interpretation bias cannot be excluded despite the use of common rules, documentary-evidence checking, and contextual cross-checking. The architecture-related findings are case-specific conceptual insights. Transferability of the proposed architecture beyond this case has not been established; the evidence-to-design reasoning process also requires beyond-case evaluation. Broader generalizability of DDD/EDA design principles is not established.

IV. CONCLUSION

This study derived a conceptual integration architecture for a banking bond-transaction process from 275 operational records. The analysis produced four bounded integration indications, six requirements, six conceptual bounded contexts, and a conceptual EDA model that distinguishes business-event/status semantics from response-dependent interaction. The resulting hybrid architecture preserves the direct instruction path and other response-dependent interactions while using selective event/status propagation between Transaction Processing and Status and Reconciliation.

This staged evidence-to-design process makes the link between operational integration evidence and the resulting responsibility, interaction, and conceptual architecture decisions inspectable. The seven-session FGD with 20 practitioners provides case-based, non-independent conceptual evaluation of that reasoning and design. The architecture remains case-specific; operational effectiveness has not been established. Transferability, independent review, proof-of-concept implementation, and non-functional assessment remain future work.

ACKNOWLEDGMENT

The author thanks the practitioners who contributed to the operational-evidence review and later FGD. Individual, organizational, and system identities are reported only within the study confidentiality controls.

DATA AVAILABILITY AND CONFIDENTIALITY

The underlying operational records, RCA documents, process/architecture artifacts, FGD notes, and supporting materials are confidential case records and are not publicly available. The manuscript reports only aggregated and anonymized representations; source provenance and record-level traceability remain in a controlled research register. Supplementary Material S1 functions as a publication-safe methodological support artifact rather than as a substitute for the confidential case records. It documents the classification, interpretation, requirement-derivation, and traceability rules, with a worked example, so the analytical procedure can be inspected and reapplied without disclosure of the source records. Where authorized access to the underlying case evidence is available, the same documented procedure can be used to reproduce the analysis. S1 alone does not permit exact reproduction or independent verification of the original 275-record findings because record-level evidence and provenance remain controlled.

REFERENCES

[1] S. Chatterjee, S. Sarker, M. J. Lee, X. Xiao, and A. Elbanna, "A possible conceptualization of the information systems (IS) artifact: A general systems theory perspective," *Information Systems Journal*, vol. 31, no. 4, pp. 550-578, 2021, doi: 10.1111/isj.12320.

[2] A. Sunyaev et al., "The future of enterprise information systems," *Business & Information Systems Engineering*, vol. 65, no. 6, pp. 731-751, 2023, doi: 10.1007/s12599-023-00839-2.

[3] T. Grisold, C. Janiesch, M. Röglinger, and M. T. Wynn, "Managing dynamics in and around business processes," *Business & Information Systems Engineering*, vol. 66, pp. 533-540, 2024, doi: 10.1007/s12599-024-00895-2.

[4] B. Guggenheim, S. Kraenzlin, and C. Meyer, "(In)efficiencies of current financial market infrastructures: An empirical assessment," *Swiss Journal*

of Economics and Statistics, vol. 158, Art. no. 19, 2022, doi: 10.1186/s41937-022-00098-0.

[5] J. A. Ruiz Ceniceros, J. A. Aguilar-Calderón, R. Espinosa, and C. Tripp-Barba, "The external and data loose coupling for the integration of software units: A systematic mapping study," *PeerJ Computer Science*, vol. 7, Art. no. e796, 2021, doi: 10.7717/peerj-cs.796.

[6] M. Aouachria, A. Leshob, A. R. Ghomari, and M. Aouache, "A process mining method for inter-organizational business process integration," *ACM Transactions on Management Information Systems*, vol. 15, no. 1, Art. no. 2, pp. 1-29, 2024, doi: 10.1145/3638062.

[7] J. Cândido, M. Aniche, and A. van Deursen, "Log-based software monitoring: A systematic mapping study," *PeerJ Computer Science*, vol. 7, Art. no. e489, 2021, doi: 10.7717/peerj-cs.489.

[8] C. Daraio, S. Di Leo, and M. Scannapieco, "Accounting for quality in data integration systems: A completeness-aware integration approach," *Scientometrics*, vol. 127, pp. 1465-1490, 2022, doi: 10.1007/s11192-022-04266-0.

[9] C. Verbruggen, A. Goossens, J. De Smedt, J. Vanthienen, and M. Snoeck, "iDOCEM: Defining a common terminology for object-centric event logging and data-centric process modelling," *Software and Systems Modeling*, vol. 24, no. 1, pp. 9-33, 2025, doi: 10.1007/s10270-024-01191-z.

[10] O. Özkan, Ö. Babur, and M. van den Brand, "Domain-Driven Design in software development: A systematic literature review on implementation, challenges, and effectiveness," *Journal of Systems and Software*, vol. 230, Art. no. 112537, 2025, doi: 10.1016/j.jss.2025.112537.

[11] C. Zhong, S. Li, H. Huang, X. Liu, Z. Chen, Y. Zhang, and H. Zhang, "Domain-Driven Design for microservices: An evidence-based investigation," *IEEE Transactions on Software Engineering*, vol. 50, no. 6, pp. 1425-1449, 2024, doi: 10.1109/TSE.2024.3385835.

[12] K. Farias and L. Lazzari, "Event-driven architecture and REST architectural style: An exploratory study on modularity," *Journal of Applied Research and Technology*, vol. 21, no. 3, pp. 338-351, 2023, doi: 10.22201/icat.24486736e.2023.21.3.1764.

[13] H. Cabane and K. Farias, "On the impact of event-driven architecture on performance: An exploratory study," *Future Generation Computer Systems*, vol. 153, pp. 52-69, 2024, doi: 10.1016/j.future.2023.10.021.

[14] P. Ataei and M. Atemkeng, "Terramycelium: a reference architecture for adaptive big data systems," *Journal of Big Data*, vol. 12, Art. no. 260, 2025, doi: 10.1186/s40537-025-01311-6.

[15] A. Singjai and U. Zdun, "Conformance assessment of architectural design decisions on API endpoint designs derived from domain models," *Journal of Systems and Software*, vol. 193, Art. no. 111433, 2022, doi: 10.1016/j.jss.2022.111433.