

Explainable Service Boundary Decision Framework for Monolith Migration Using DDD and Quantitative Metrics

Irgi Achmad Fauzi, Muhammad Yusril Helmi Setyawan*, Mohamad Nurkamal Fauzan

Applied Bachelor Program of Informatics Engineering, School of Information Technology, Universitas Logistik dan Bisnis Internasional, Bandung, Indonesia

Email: ¹irgifuuzi13@gmail.com, ^{2,*}yusrilhelmi@ulbi.ac.id, ³m.nurkamal.f@ulbi.ac.id

Correspondence Author Email: yusrilhelmi@ulbi.ac.id*

Submitted: 20/07/2026; Accepted: 11/09/2026; Published: 30/09/2026

Abstract—Identifying a service boundary is a critical decision in monolith-to-microservices migration. Intuition alone is hard to reproduce, while technical metrics without domain context can overlook business semantics. This study develops an explainable boundary-decision framework linking Domain-Driven Design (DDD), implementation evidence, and quantitative metrics for the FemaleFit e-commerce monolith. Domain mapping traced business capabilities to endpoints, controllers, models, and tables, producing 13 initial contexts and seven atomic candidates. Candidates were scored with M01–M13 under three weighting schemes (W1–W3), a practical-tie rule, and boundary sensitivity analysis. C01 Identity & Access and C03 Craft Catalog formed a practical tie, differing by 0.002803 under W1. C03 was selected on the judgement that it had clearer domain alignment, an explicit routing seam, a verified core flow, and a more contained blast radius. It was extracted as a Catalog Service behind a Strangler Gateway, with 34 unique HTTP method–path pairs reconciled against the migration contract. Functional evaluation covered one representative case for each of the 34 frozen Catalog contracts, including negative/error cases. All 540 retained read requests completed without failure across three independent runs per condition; latency findings were descriptive and mixed. The metrics indicate lower structural coupling and process-level deployment independence, but not database independence. Findings are bounded by one case, one service, a sequential local workload, and a shared database.

Keywords: Domain-Driven Design; Explainable Decision; Microservices; Quantitative Metrics; Service Boundary

1. INTRODUCTION

Monolithic architecture remains relevant in the early stages of development because application functions can be built, tested, and deployed as a single unit. This form reduces inter-service coordination, simplifies transactions, and allows small teams to move quickly. As the system grows, added features, dependencies, data, and teams make local changes liable to affect other parts, regression testing widens, and the whole application must be redeployed. Habib et al. showed that migration requires architectural and organizational readiness, not merely code separation [1]. Ayas et al. found that the move toward microservices involves technical and systemic changes that are difficult to separate [2].

Microservices offer services with focused responsibilities and process-level deployment, which can support selective scaling, change isolation, and team autonomy. These benefits are not obtained automatically. Migration adds API contracts, network communication, observability, data consistency, failure modes, and new operational costs. Determining the service boundary is therefore a central concern. A boundary that is too large preserves coupling and deployment friction, while a boundary that is too small increases communication, transaction coordination, and runtime complexity. The systematic review by Oumoussa and Saidi reports that microservice identification draws on business artifacts as well as static, dynamic, and semantic analysis and optimization, but that the techniques and their evaluation remain diverse [3].

Domain mapping is inadequate when it merely follows table names, folders, controllers, or route prefixes. Such artifacts are implementation hints, not definitions of business capability. DDD requires linking objectives, processes, ubiquitous language, aggregates, ownership, and inter-context relationships. Khononov describes a bounded context as a boundary of model and domain-language consistency that must be tied to business strategy [4]. Action research by Özkan et al. reports the benefits of DDD-based refactoring in an industrial context, while also showing that its outcomes depend on organizational context and human judgment [5].

Boundary decisions that rely solely on an architect's intuition are difficult to reproduce because the underlying evidence, trade-offs, and assumptions are often undocumented. However, using metrics without domain interpretation is also inadequate. Al-Debagy and Martinek formed decompositions based on dependencies [6], while Cromlech optimizes operations, entities, cohesion, communication, and replication [7]. Kinoshita and Kanuka extended this direction through multi-objective optimization [8]. These studies demonstrate the benefits of quantitative comparison while leaving a need to interpret scores in light of the domain and migration risk.

Approaches based on business artifacts attempt to preserve semantics that code structure does not always capture. Oumoussa and Saidi applied natural language processing, semantic analysis, and clustering to business process models [9]. Maharjan et al. built a dependency graph through static analysis, extracted features with a

variational autoencoder, and clustered classes using C-means on the Food-to-Go benchmark [10]. Yılmaz and Buzluca developed a fuzzy logic-based quality model to flag microservice candidates with low maintainability [11]. This literature enriches automation and evaluation, but the relationship between business rationale, code and database evidence, weighting sensitivity, and boundary implementation still needs to be shown in a more integrated way on empirical cases.

Architectural decisions also require rationale documentation that can be inspected. The study by Buchgeher et al. on open-source repositories shows that Architecture Decision Records can capture decisions and their rationale in relatively lightweight artifacts, although adoption remains limited [12]. The term explainable in this study does not refer to Explainable AI; it refers to the ability to connect a boundary decision with business evidence, code, data, metrics, sensitivity analysis, risk, and migration validation results.

Incremental migration reduces risk compared with a simultaneous replacement. Newman discusses evolutionary monolith-to-microservices patterns, including the use of seams and the Strangler Fig to move functions incrementally [13]. Hassan et al. proposed a pattern-based automated migration framework [14], while Nordli et al. validated a cloud-native approach through two proofs of concept [15]. On the data side, Kazanavičius et al. showed that splitting a monolith database toward polyglot persistence requires its own stages and transformations [16]. These findings support using a shared database as a transition strategy, but not as evidence of database independence.

The impact of migration cannot be inferred from code-structure changes alone. X. Li et al. report the importance of ownership and organizational coupling in microservice projects [17]. Faustino et al. found that stepwise migration requires substantial effort and may be accompanied by performance changes during the transition [18]. After accounting for confounding factors, Saarimäki et al. found no significant velocity difference between microservice and monolith project cohorts [19].

The literature has discussed DDD-based decomposition, quantitative boundary identification, migration frameworks, incremental extraction patterns, performance evaluation, and architectural decision traceability. These approaches show that DDD can be connected to model mapping, databases, and service extraction priorities. However, in the studies reviewed here, those elements are generally still discussed in separate threads. This study therefore focuses on integrating Business Capability → Source-Code Evidence → Database Evidence → DDD Boundary → Quantitative Metrics → Ranking → Sensitivity → Migration Validation into a single audit chain. The targeted gap is not the absence of each element, but the lack of a consistent connection among domain reasoning, quantitative scoring, decision traceability, sensitivity analysis, implementation, and before–after evaluation on a single boundary decision.

Based on this gap, this study constructs and evaluates an explainable service boundary decision framework for the migration of the FemaleFit e-commerce monolith. The framework links DDD mapping based on business, source-code, and database evidence with M01–M13 extractability scoring, ranking, practical tie, sensitivity, and migration risk. One boundary is then extracted through a Strangler Gateway and evaluated in terms of functional equivalence, runtime, and architectural impact. The contribution is limited to integration and empirical validation in a single case; the manuscript claims neither absolute novelty nor generalizable superiority.

2. RESEARCH METHODOLOGY

2.1 Research Stages

This study uses an experimental case study of the FemaleFit e-commerce application. The procedure consists of ten connected stages: (1) understanding the system and business; (2) verifying the monolith baseline; (3) mapping the domain; (4) forming boundary candidates; (5) extracting metrics; (6) normalizing values and computing the Extractability Score; (7) ranking and sensitivity analysis; (8) extracting the selected service; (9) evaluating the conditions before and after extraction; and (10) auditing validity and reproducibility. Artifacts from each stage feed the next stage, so the basis of the decision remains traceable.

Figure 1 shows the stage order and the artifact handover between consecutive stages.

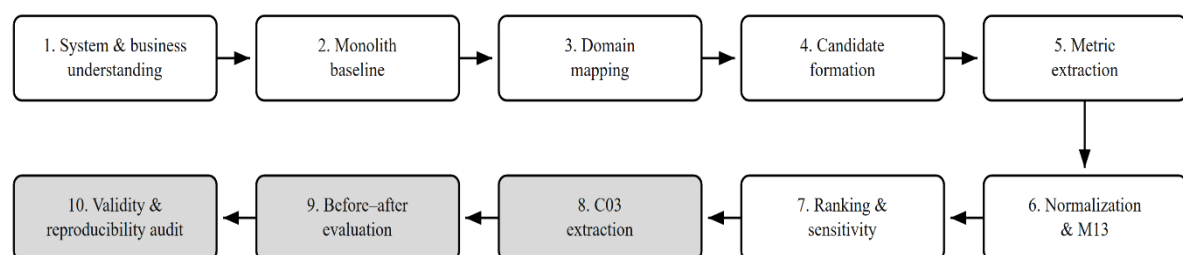


Figure 1. Research stages

2.2 Object and Data Sources

The research object is FemaleFit, an e-commerce application for women's sports products with a Go/Fiber backend and MySQL database. The verified baseline comprises the active code, 31 tables, 304 columns, 31 primary keys, 26 foreign keys, and ten critical smoke-test scenarios. It provides the before–after comparator, not evidence of exhaustive system testing. Experiments ran locally with Go 1.25 for Windows 64-bit, MySQL 8.4.3, 16 logical CPUs, and loopback communication. Memory capacity was not recorded; CPU and memory utilization were therefore not analyzed.

Evidence comprised source code, HTTP endpoints, controllers/functions, models, tables, foreign keys, and documented business knowledge. The dataset contains candidate and traceability matrices, operation and dependency manifests, metric and weighting results, route mappings, functional-equivalence scenarios, runtime measurements, and architectural comparisons. Endpoints were traced to their invoked controller/function and model access to table operations and dependencies. Credentials, tokens, .env contents, authentication payloads, and real user data were excluded from publication material.

After the initial mapping was frozen, one external reviewer, a backend developer with experience in Go, REST API development, MySQL database design, and e-commerce application development, assessed 32 traceability records and the Catalog–Inventory and Ordering–Payment boundary alternatives. The reviewer had no involvement in FemaleFit development, the initial mapping, candidate selection, or experimental execution. Each record linked a capability and process to its endpoint, controller, model, table, proposed candidate boundary, and DDD element. The reviewer recorded AGREE, PARTIALLY_AGREE, DISAGREE, or INSUFFICIENT_EVIDENCE, together with confidence and comments. The reviewer declared no financial or supervisory conflict of interest.

Because Firebase credentials were unavailable, Notification/Firebase was replaced locally by a best-effort no-op research adapter. It affects only the non-critical notification path after Craft creation and does not alter the main Craft response or the evaluated Category, Image, Size, and Stock data relationships. The results therefore do not claim Notification equivalence, Firebase delivery, or production notification capability.

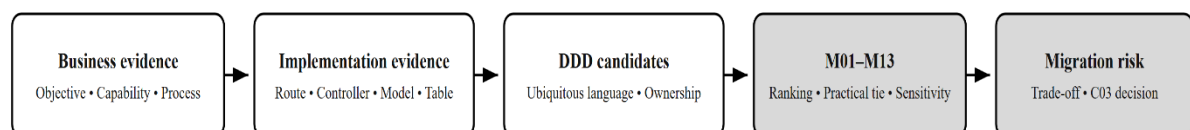
2.3 Domain Mapping

Domain mapping was performed before metrics were computed; candidates were not derived directly from formulas. The analysis follows the chain Business Objective → Business Capability → Business Process → Endpoint → Controller → Model → Table → Dependency → Candidate Bounded Context. Functions were recorded as implementation evidence linking controllers to model operations, but the location of a function in a particular file was not used as a basis for boundaries. Folder names, route prefixes, and table proximity were likewise treated as initial hints, not as sufficient evidence for establishing a bounded context.

Artifacts were grouped by considering shared business objectives and processes, cohesion of responsibilities, dependency patterns, clarity of data ownership, ubiquitous language, and fit with the bounded-context hypothesis. Business terms still in active use were distinguished from legacy terms. Evidence IDs were used to record relationships between candidates, possible aggregate roots and invariants, and tables that are written or only read. This procedure produced the initial candidates; only candidates with sufficiently clear capabilities and adequate implementation evidence proceeded to quantitative comparison. The use of DDD in this study follows the bounded-context principle [4], and is informed by empirical evidence on DDD-based refactoring [5].

The researcher-coded mapping was retained as the primary traceability record and was not overwritten by external review. Agreements supported mapped responsibilities, whereas partial agreements were retained as explicit challenges to candidate membership, ownership, and boundary alternatives. These challenges were not used to recalculate M13 after ranking; instead, they were interpreted alongside the alternative-boundary and leave-one-metric-out analyses.

Figure 2 traces a single decision from business objective to migration validation and marks the point at which the metric stage enters the chain.



Explainable = traceability of the architectural decision, not Explainable AI

Figure 2. The explainable service boundary decision chain

Source: evidence chain and decision rationale of the final experiment.

2.4 Quantitative Metrics

The main unit of analysis is a candidate bounded context with an identified set of operations, artifacts, tables, and dependencies. M01–M08 and M12 are grouped as technical metrics; M09–M11 as domain metrics; M13 is a

composite score. Table 1 presents the operational definitions used. The notation O_c denotes the candidate's operations; A_c the atomic units composing the candidate; T_ext , T_all , T_cross , T_access , $T_written$, and $T_exclusive_written$ respectively denote the sets of external dependency targets, all dependency targets, cross/shared tables, accessed tables, written tables, and written tables with an exclusive owner. E_ii , E_io , and E_oi are internal, outgoing, and incoming edges; N_in and N_out are candidate neighbors; while FK_cross and $FK_incident$ are cross-boundary foreign keys and all incident foreign keys.

The metrics represent complementary constructs rather than interchangeable indicators. M01 and M06 represent internal responsibility concentration and cohesion; M02–M05 and M12 represent structural dependency; M07–M09 represent data-boundary conditions; and M10–M11 represent semantic and bounded-context alignment.

The metric set is a transparent, case-specific decision protocol rather than a universal measurement scale. Each value is traceable to a frozen manifest, raw numerator and denominator, evidence reference, direction rule, and completeness status. M10 and M11 require business, implementation, and data evidence; incomplete evidence produces NA and blocks M13 [20].

Table 1. Concise operational definitions of M01–M12

ID	Metric; unit/purpose	Raw formula	Direction & normalization	Source/NA
M01	Internal Dependency Coverage; operations	$ \{o \in O_c: \text{internal_dependency_count}(o) > 0\} / O_c $	$\uparrow$; $p01=M01$	operation/dependency/ownership; incomplete $\rightarrow$ NA
M02	External Dependency Ratio; target	$ T_ext / T_all $	$\downarrow$; $p02=1-M02$	canonical targets/ownership; unresolved $\rightarrow$ NA
M03	Structural Coupling Ratio; edge	$(E_io + E_oi)/(E_ii + E_io + E_oi)$	$\downarrow$; $p03=1-M03$	static call graph; incomplete edge $\rightarrow$ NA
M04	Normalized Afferent Coupling; adjacency	$ N_in / (13- A_c)$	$\downarrow$; $p04=1-M04$	inbound adjacency; unresolved member $\rightarrow$ NA
M05	Normalized Efferent Coupling; adjacency	$ N_out / (13- A_c)$	$\downarrow$; $p05=1-M05$	outbound adjacency; unresolved target $\rightarrow$ NA
M06	Internal Cohesion; operation pairs	related unordered pairs / $C(O_c , 2)$	$\uparrow$; $p06=M06$	owned element/invariant; $ O_c < 2 \rightarrow$ NA
M07	Database Dependency Ratio; tables	$ T_cross / T_access $	$\downarrow$; $p07=1-M07$	table access/ownership; no access $\rightarrow$ NA
M08	Foreign-Key Dependency Ratio; incident FKs	$ FK_cross / FK_incident $	$\downarrow$; $p08=1-M08$	26-FK inventory; incomplete $\rightarrow$ NA
M09	Data Ownership Clarity; written tables	$ T_exclusive_written / T_written $	$\uparrow$; $p09=M09$	write/owner manifest; no writes $\rightarrow$ NA
M10	Business Capability Alignment; artifacts	evidence_supported_aligned / all_classified	$\uparrow$; $p10=M10$	business+implementation evidence; ambiguity $\rightarrow$ NA
M11	Bounded-Context Alignment; boundary	$\Sigma(r_1 \dots r_6) / 6$	$\uparrow$; $p11=M11$	binary rubric/Evidence ID; any unassessed $\rightarrow$ NA
M12	Structural Independence; operations	$ \{o \in O_c: \text{external_dependencies}(o)=0\} / O_c $	$\uparrow$; $p12=M12$	code/data/system manifest; unresolved $\rightarrow$ NA

Source: formula catalog, amendment v1.1, and the final experiment metric dataset. M13 is a composite score, not a raw metric.

M01, M06, M09, M10, M11, and M12 have a positive direction. M02, M03, M04, M05, M07, and M08 have a negative direction because lower raw values are more favorable. The computation uses distinct targets or edges so that repeated calls within long functions do not dominate. M06 uses unordered pairs that share a candidate-owned domain element or the same aggregate invariant. M10 follows protocol rule version 1.1: alignment must be supported by consistent business/process and implementation/data evidence. The researcher-coded values were subsequently examined through an independent domain-validation layer; reviewer judgements were reported separately and, where they materially differed, evaluated as an alternative coding scenario rather than silently replacing the frozen primary dataset. M11 sums six binary criteria (language, process objective, aggregate

consistency, data ownership, explicit relationships/interfaces, and business confirmation) and divides the total by six.

A zero value is valid only when the scan is complete and the denominator is valid. Specifically, a candidate with no incident edges in M03 or no incident FKs in M08 may score zero only with the corresponding completeness flag. M06 becomes NA with fewer than two operations; M07 becomes NA without accessed tables; M09 becomes NA without written tables; and any unresolved dependency, ownership, membership, or M11 criterion yields NA. There is no imputation or weight renormalization: one missing M01–M12 value blocks M13 and the ranking of that candidate. Detailed definitions of numerators, denominators, data sources, interpretation, and missing-value handling are provided in the accompanying audit material.

2.5 Normalization and M13

All M01–M12 values are already ratios in [0,1]; therefore, the study does not use sample min–max normalization. For benefit and cost metrics, the transformations are expressed in (1) and (2), respectively.

$$p_i c = raw_i c \quad (1)$$

$$p_i c = 1 - raw_i c \quad (2)$$

The technical and domain composites are computed with (3) and (4), with M12 included in the technical group.

$$T(c) = (p_{01} + p_{02} + p_{03} + p_{04} + p_{05} + p_{06} + p_{07} + p_{08} + p_{12}) / 9 \quad (3)$$

$$D(c) = (p_{09} + p_{10} + p_{11}) / 3 \quad (4)$$

Under W1, each metric receives an equal weight. Defining $m_i(c) = p_i(c)$, the weights and M13 are computed using (5) and (6).

$$w_i = 1 / 12 \quad (5)$$

$$M13W1(c) = \sum_{i=1}^{12} w_i \times m_i(c) \quad (6)$$

The formulas, denominators, directions, and weights were fixed before ranking and applied consistently to all candidates. Within this case-specific protocol, M13 is a prespecified comparison rule, not a universal instrument for service-boundary selection, a probability of migration success, an absolute quality score, or an automatic decision [21].

2.6 Sensitivity Analysis

The sensitivity analysis tests the decision's dependence on weighting and boundary variation. W1 assigns a weight of 1/12 to each of M01–M12. W2 emphasizes technical evidence, while W3 emphasizes domain/DDD evidence:

$$W2(c) = 0.80 T(c) + 0.20 D(c) \quad (7)$$

$$W3(c) = 0.40 T(c) + 0.60 D(c) \quad (8)$$

W1 is the primary neutral baseline because it assigns equal weight to all twelve complete metrics. W2 and W3 are not presented as expert-derived weights. They are deliberately contrasting stress-test scenarios: W2 assigns 80% to technical evidence to test whether a technically dominant interpretation changes the baseline order, whereas W3 assigns 60% to domain/DDD evidence to test whether the decision changes when semantic boundary evidence is given greater emphasis. The two scenarios therefore bracket W1 with materially different, but non-exclusive, technical and domain priorities. Their purpose is to test ranking sensitivity rather than to claim that 0.80/0.20 or 0.40/0.60 are universally optimal proportions.

All schemes use the same normalized matrix without renormalization. Besides weight changes, the analysis tests a Catalog–Inventory split and an Ordering–Payment merge. An absolute W1 difference below 0.01 was fixed before the decision as a practical materiality rule on the [0,1] score scale; it is not a statistical significance bound. Because no universal threshold exists for this protocol-specific composite, the study also evaluates how the practical top set changes under alternative thresholds. Candidates in the practical-top set are compared using W2, W3, domain evidence, alternative boundaries, and migration risk. The top rank therefore does not directly determine the selected boundary.

2.7 Migration Implementation

C03 Craft Catalog was selected only after domain mapping, metric computation, ranking, sensitivity, and risk assessment were complete. The implementation uses a Strangler Gateway with an allowlist of exact HTTP method–path pairs. Selected routes are forwarded to the Catalog Service, while other routes, including the prefix-overlapping Checkout and Review functions, continue to the monolith. This strategy supports cutover and rollback at the routing layer. Both processes still use the same physical database as a transition strategy; process-level deployment independence is not interpreted as database independence. The incremental approach and the migration seam follow incremental migration principles [13], while the consequences of the shared database are assessed separately [16].

2.8 Evaluation

The evaluation covers four dimensions. First, the boundary-selection metrics M01–M13 support the boundary decision. Second, the runtime evaluation uses p50 and p95 obtained with the nearest-rank procedure, together with success rate and error rate. The monolith and transition conditions were run on the same machine, gateway origin,

database, fixtures, and request order. The runtime evaluation uses sequential loopback requests only. This design supports a controlled local before–after comparison but does not represent production-level microservice workloads, concurrent users, distributed-network latency, autoscaling, or contention. The reported latency results must therefore not be interpreted as production-capacity evidence. For each read workload, three warm-up requests were discarded and 30 requests were retained per run. The runtime evaluation was repeated in three independent runs per condition, with the monolith and migrated conditions alternating execution order across runs. Run-level p50 and p95 values, sample standard deviations, and two-sided 95% confidence intervals were calculated from the three runs. The monolith control binary uses the baseline business source with an added port argument and a no-schedulers control to prevent cron mutations, so the measurement binary is not byte-identical to the initial baseline binary.

Third, the architectural evaluation compares dependency targets, structural coupling, structural independence, deployment independence, and data ownership. Fourth, functional equivalence was expanded from the initial ten critical scenarios to one representative case for each of the 34 frozen Catalog contracts. Eleven representative read cases used existing data, while 23 non-destructive negative/error cases used malformed JSON, empty required input, missing query parameters, or nonexistent identifiers. All evaluated migrated requests were routed to the Catalog service and produced the same HTTP status, content type, and response body as the monolith. A separate disposable lifecycle test verified Craft creation, image upload, size/stock creation, update, detail retrieval, deletion, and fixture cleanup through the migrated route. This coverage does not claim equivalence for every valid payload or external-service variation. CPU and memory utilization were not measured and are absent from both the analysis and the claims.

2.9 Validity Controls

The validity controls include fixing formulas and weights before ranking, applying the rules consistently, verifying the baseline, and tracing key claims to business, code, database, metric, and evaluation evidence. Rank changes due to weights or boundaries are reported through the sensitivity analysis. A reproducibility audit then re-examines the normalization, scores, ranking, sensitivity, route mapping, and scenario sameness based on the stored dataset.

The Notification/Firebase adapter is treated as both an environmental control and a validity threat. The adapter removes the need for credentials and external services at startup, but it also removes the Notification/Firebase runtime targets from the extracted process. A three-configuration static comparison was therefore used to distinguish the extraction effect from the adapter effect: (A) the monolith with the direct Notification dependency, (B) an extracted Catalog service retaining that dependency, and (C) the extracted Catalog service using the research no-op adapter. M02, M03, and M12 are unchanged from A to B; the change from B to C is consequently attributed to containing the Notification/Firebase dependency through the adapter. This comparison does not demonstrate real Firebase delivery or production Notification capability.

Business ground truth was first obtained from the researcher's documented system knowledge and triangulated with business-process records, source code, endpoint-controller mappings, database mappings, and foreign-key evidence. Because the researcher also acted as the system developer, these initial sources did not by themselves constitute independent validation; their domain interpretation was subsequently subjected to the external review described in Section 2.2. Other validity threats are the single-machine environment, sequential loopback workload, shared database, and the limited number of repeated runtime runs.

The reproducibility material contains the formula catalog, raw and normalized matrices, weight contributions, rankings, 63 sensitivity results, the route-reconciliation map, equivalence scenarios, and raw runtime data. Based on the stored data, the audit reproduced 84/84 normalized values, rankings for 7/7 candidates, and 63/63 sensitivity results. The supplementary material additionally contains the 34-contract equivalence results, repeated-runtime raw data and statistical summary, and the adapter-effect comparison. The complete reproducibility package is available at (<https://github.com/irgifuauzi/femalefit-service-boundary-reproducibility>).

3. RESULTS AND DISCUSSION

3.1 The FemaleFit Monolith Baseline

Before extraction, FemaleFit ran as a controller-oriented monolith: centralized route registration forwards requests to middleware, controllers, and integration services, while every capability accesses a single femalefitta database. The verified schema contains 31 tables, 304 columns, 31 primary keys, and 26 foreign keys. Craft and Category media are stored in a local directory, so the application code, shared database, and file storage form one deployment unit.

Endpoint mapping links every HTTP method–path pair to the invoked controller/function, model, table operations, and dependencies. In this way, a route or controller name is not directly treated as a boundary. At baseline, Catalog was still a logical boundary within the monolith process: Catalog reads Identity/Profile, Promotion, and Review data; Cart, Ordering, Chat, and Promotion read Catalog tables directly; and Ordering also modifies Stock in craft_size. These findings show that feature proximity at the interface does not by itself indicate independent code and data ownership. The frozen baseline was verified using the same database state and core

Craft flow; data counts and checksums returned to their initial state after testing. The subsequent 34-contract evaluation is reported in Section 3.9.

3.2 Applied Domain Mapping

Applying the procedure in Section 2.3 to FemaleFit produced the 13 initial contexts reported in Section 3.3. The mapping linked each context to its business objectives, capabilities, processes, endpoints, controllers/functions, models, tables, and dependencies. Candidate assignment required convergent business and implementation evidence; the resulting contexts and their ranking eligibility are reported next.

3.3 Service Boundary Candidates

The mapping produced 13 initial contexts, C01–C13. Seven of them had sufficient active evidence to be treated as atomic candidates and enter the ranking: C01 Identity & Access, C02 User Profile, C03 Craft Catalog, C05 Shopping Cart, C06 Ordering & Checkout, C07 Payment, and C08 Shipping & Fulfilment.

Six other contexts were still recorded so that relevant capabilities were not lost from the analysis, but they were not included in the main ranking. C04 Inventory is used as a boundary variant to test separating Size/Stock from Catalog. C09 Promotion, C11 Address & Geography, and C13 Refund & Cancellation are secondary or preliminary. C10 Communication and C12 Notification are static/preliminary because their implementation evidence or business rules were insufficient for comparison in the main stratum. Thus, 13 refers to the number of initial contexts analyzed, not the number of microservices built.

C03 illustrates this grouping. Craft is the main product concept; Category provides the Catalog taxonomy; Image is part of product presentation; while Size and Stock form availability variations within the main boundary under test. These five elements form Craft Catalog because they support the processes of defining, publishing, discovering, and presenting products. The evidence is concentrated in Craft/Category/Size operations, two main controllers, five models/tables proposed for logical ownership, and consistent business language.

Proximity to Craft does not automatically yield Catalog ownership. Checkout uses Craft as input to form an Order and therefore belongs to Ordering & Checkout. Payment manages Midtrans interactions and payment status on transactions; this capability remains separate even though its flow uses product prices. Profile only supplies maker identity or display data, while Review has its own lifecycle and craft_ulasan data. The routes GET /api/craft/:id/checkout and GET /api/craft/ulasan/rerata show that sharing the /api/craft/* prefix is not sufficient to move them into Catalog.

Table 2 lists the 13 initial contexts with their business capabilities and status.

Table 2. Service candidates and business capabilities

ID	Candidate/context	Business capability	Status
C01	Identity & Access	Register, authenticate, verify, authorize	Ranked
C02	User Profile	Maintain buyer profile	Ranked
C03	Craft Catalog	Define, publish, discover, present products	Selected
C04	Inventory	Manage independent Size and Stock	Variant
C05	Shopping Cart	Persist and revise purchase intent	Ranked
C06	Ordering & Checkout	Create and progress an Order	Ranked
C07	Payment	Initiate and reconcile payment	Ranked
C08	Shipping & Fulfilment	Quote, create, track delivery	Ranked
C09	Promotion	Manage incentives and campaigns	Secondary
C10	Communication	Manage product conversations	Preliminary
C11	Address & Geography	Maintain destination/regional lookup	Secondary
C12	Notification	Persist and deliver notifications	Preliminary
C13	Refund & Cancellation	Cancel and compensate an Order	Secondary

Source: domain mapping and candidate inventory derived from the final experimental system.

External review of the 32 traceability records returned 24 AGREE and eight PARTIALLY_AGREE decisions; no DISAGREE or INSUFFICIENT_EVIDENCE decisions were recorded. All five C03 Craft Catalog records were agreed with HIGH confidence. The partial decisions challenged C04 as an independently owned current boundary, favored the merged Ordering–Payment interpretation (ORD-ALT-B) for the two C07 Payment records, and noted preliminary or shared ownership concerns for Shipping, Promotion, and Refund. Thus, the review supports the C03 mapping while retaining the alternative-boundary uncertainties in the sensitivity analysis.

3.4 Metric Computation

The dataset produced 84 raw M01–M12 values for the seven atomic candidates. In the audit material, each value has traceable evidence and denominators. M01–M08 and M12 represent technical evidence, while M09–M11 represent domain evidence. Benefit directions were applied before the composites were computed: higher values favor M01, M06, and M09–M12, while lower values favor M02–M05, M07, and M08. The full matrix is provided as supplementary material; this section discusses the patterns most relevant to the decision.

Under the defined metrics, C03 combines complete internal dependency coverage and full domain alignment (M01, M09–M11 = 1.000000) with the highest observed operational independence (M12 = 0.823529). Its extraction costs remain material: M02 = 0.833333, M03 = 1.000000, and M08 = 0.444444. C03 is therefore not coupling-free.

The remaining candidates exhibit different trade-offs in cohesion, data dependence, foreign-key coupling, and operational independence. Detailed profiles and the complete matrix are available in the reproducibility package. M10 remains constant at 1.000000 across candidates and is consequently addressed through the leave-one-metric-out check below.

All 84 M01–M12 values were complete and traceable to raw evidence. M10 equals 1.000000 for all seven ranked candidates and therefore does not individually distinguish the primary ranking. Leave-one-metric-out checks showed that omitting either M10 or M11 retained the W1 order C01, C03, C05, C02, C07, C08, and C06. The full rubric, evidence register, and robustness calculations are provided as supplementary material.

3.5 Candidate Ranking

Under equal weighting W1, C01 ranks first with a score of 0.733498 and C03 second with 0.730695. The difference of 0.002803 is below the practical-tie threshold of 0.01, so both entered the practical top set rather than being treated as a materially distinct winner and runner-up. C05 follows at rank 3 with 0.722222, then C02, C07, C08, and C06.

When the technical group is weighted 0.80 in W2, C01 and C03 remain at ranks 1 and 2; C03's score is 0.712742. Under W3, which weights domain/DDD evidence 0.60, C03 moves to rank 1 with 0.856371 and C01 to rank 4. This shift shows that C03's position is influenced by the emphasis on domain evidence. The results do not support a generalization that C03 is always the highest-scoring candidate.

3.6 Sensitivity Analysis

Changing the boundary definition had a larger impact than changing the weights. In CAT-ALT-2, Size/Stock is separated into Inventory; Craft Catalog without Inventory drops to rank 5 in W1 with a score of 0.684232, while the hypothetical Inventory variant ranks first. This variant was not selected because its independent business capability had not been validated and Stock control was still spread as far as Ordering. Based on its stability under weight changes but sensitivity to the Inventory split, the C03 decision is classified as MODERATE_STABILITY.

To extend the three prespecified schemes, the technical weight was varied from 0.0 to 1.0 in increments of 0.1, with the domain weight set to 1 minus the technical weight. C03 ranked first from technical weights of 0.1 to 0.7. C01 ranked first when the technical weight was 0.8 or higher. Under a purely domain-weighted scenario, C02, C03, and C05 obtained the same domain score and therefore formed a domain-only top set. The result shows that C03 is favored when domain evidence is given substantial, but not exclusive, emphasis; it is not the universal numerical leader under all possible weights.

Threshold sensitivity also shows that the C01–C03 relationship depends on the declared materiality rule. Their W1 difference is 0.002803. Under a threshold of 0.001, C01 is the sole numerical leader; under thresholds of 0.003, 0.005, and 0.010, C01 and C03 form the practical top set; and under 0.020, C05 also enters the set. The 0.01 rule is therefore retained as a predeclared decision-support convention rather than interpreted as an empirically universal cutoff. The final selection of C03 consequently relies on the complete evidence chain and migration-risk assessment, not on the threshold alone.

The W1–W3 summary in Table 3 shows the C01–C03 practical tie and the change in C03's position when the domain weight is increased.

Table 3. Ranking and sensitivity analysis

ID	W1 (rank)	W2 (rank)	W3 (rank)	Interpretation
C01	0.733498 (1)	0.723139 (1)	0.806014 (4)	<i>Practical tie</i>
C03	0.730695 (2)	0.712742 (2)	0.856371 (1)	Selected
C05	0.722222 (3)	0.703704 (3)	0.851852 (2)	Eligible
C02	0.702381 (4)	0.682540 (4)	0.841270 (3)	Eligible
C07	0.625694 (5)	0.630370 (5)	0.592963 (7)	Not prioritized
C08	0.605266 (6)	0.597469 (6)	0.659846 (5)	Not prioritized
C06	0.588889 (7)	0.586667 (7)	0.604444 (6)	Not prioritized

Source: final experiment W1–W3 rankings. CAT-ALT-2 lowers Craft Catalog without Inventory to W1 rank 5 and underlies the MODERATE_STABILITY status.

3.7 An Explainable Service Boundary Decision

The C03 decision was built through the chain Business Objective → Product Catalog Capability → Craft-related Processes → 34 Method–Path Operations → Controllers → Models → Tables → Craft Catalog Boundary → M01–M13 → Ranking → Sensitivity → Migration Risk → Decision. This chain helps distinguish factual evidence, domain interpretation, and migration considerations. M13 is used to filter candidates worth comparing, not to replace boundary or risk judgment.

C03 was selected from the practical top set after the scores and migration risk were read together. Its W1 difference from C01 is only 0.002803, but C03's evidence is more concentrated in the Product Catalog capability. The Catalog core flow also has a repeatable functional baseline, and the route seam can be formulated as a method–path allowlist with explicit exceptions for Checkout and Review. Based on these characteristics, C03's migration blast radius is judged more manageable than Identity & Access, which is used across Profile, Cart, Ordering, Communication, and Notification. C03's position under W3 reinforces the domain considerations, while the W1 and W2 results are retained as part of the decision.

The final boundary covers 34 unique HTTP method–path pairs: 33 Catalog operations in the Craft route group and one Category operation at /api/master/kategori/shopbykategori. The number expresses the breadth of the route contract, not the number of boundaries or microservices. Only C03 Craft Catalog was selected for extraction. The method–path-based scope makes the decision testable: operations in the allowlist are routed to the extracted service, while other routes remain handled by the monolith during the transition.

3.8 Craft Catalog Extraction

The implementation extracts the single selected boundary, C03 Craft Catalog, into a separate Go/Fiber process. Its scope covers Craft as the main product concept, Category as the taxonomy, Image as product media, and Size and Stock as availability variations under the CAT-ALT-1 boundary. The five tables under Catalog's logical ownership are craft, craft_gambar, master_kategori_craft, master_size, and craft_size. This logical ownership does not yet mean full physical ownership because the Catalog Service and the monolith are still connected to the same femalefitta database.

The Strangler Gateway sits in front of both processes while preserving the public URLs. The gateway forwards the method–path pairs listed in the allowlist to the Catalog Service, while Identity, Profile, Cart, Checkout, Payment, Shipping, Promotion, and other capability routes continue to the remaining monolith. Two routes with seemingly Craft-related prefixes are treated as explicit exceptions: GET /api/craft/:id/checkout remains the responsibility of Ordering & Checkout, while GET /api/craft/ulasan/lerata remains under Review. These exact rules prevent an /api/craft/* wildcard from expanding the boundary without domain grounds.

The transition retains shared database and media storage, while non-allowlisted routes fall back to the monolith and rollback can restore Catalog traffic to it. Notification/Firebase is represented by a research no-op adapter; therefore, the evaluation does not claim production Notification or Firebase delivery.

In the Catalog Service, direct calls to Notification/Firebase are replaced with a best-effort no-op research adapter. This change removes the startup blocker when Firebase credentials are unavailable and limits the observed runtime dependencies of C03; there was no connection to production Firebase during the experiments. The adapter preserves only the non-critical flow after Create Craft, while HTTP status and Craft data changes do not depend on the notification result. The functional evaluation excludes Notification delivery, and the results are not used to claim its equivalence.

Figure 3 contrasts the monolith baseline with the transition architecture that was actually implemented. A full target microservices architecture is not depicted, because it was not produced by this study.

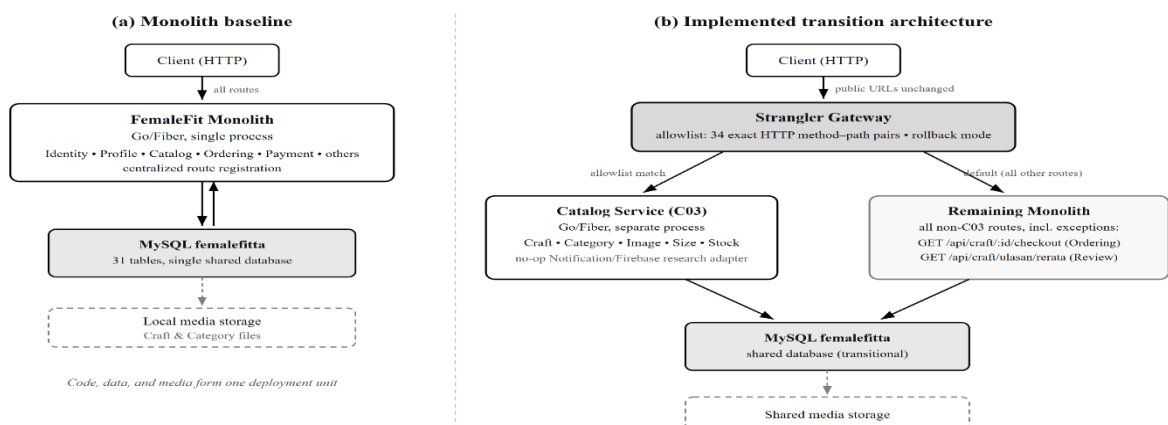


Figure 3. Monolith architecture and the C03 transition architecture

Source: verified baseline and deployment topology of the final experiment.

3.9 Functional Equivalence

Functional equivalence was initially assessed using ten critical scenarios: one authentication scenario retained in the monolith and nine Catalog scenarios routed to the new service. This evaluation was subsequently expanded to cover one representative case for each of the 34 frozen Catalog contracts. Eleven representative read cases used existing data, whereas 23 non-destructive negative/error cases used malformed JSON, empty required input, missing query parameters, or nonexistent identifiers. All 34 migrated requests carried the X-Strangler-Target: catalog-service header and produced the same HTTP status, content type, and response body as the corresponding monolith request.

A separate disposable lifecycle test created a uniquely named Craft, uploaded an image, added a size/stock record, updated and re-read the Craft, and deleted the fixture and associated test artefacts. During the initial extended evaluation, five read responses exposed a legacy wire-format mismatch in nested MasterSize fields. The Catalog service was revised to preserve the established monolith keys IDUserUpdate, DateUpdate, and Status; the final evaluation then achieved exact response-body equality for all 34 evaluated contract cases. The result covers representative success and error cases, but not every possible valid payload, media-file variation, or external-service interaction.

3.10 Runtime Evaluation

The runtime experiment was repeated in three independent runs per condition. Each run executed three discarded warm-ups followed by 30 retained sequential requests for Category list, Craft list, and Craft detail (ID 67). The order of the monolith and migrated conditions alternated across runs. This produced 540 retained observations in total. All retained requests returned HTTP 200, and migrated requests were confirmed by the gateway as targeting catalog-service.

Table 4. Repeated before-after latency results across three independent runs

Endpoint/scenario	Monolith p50, mean $\pm$ 95% CI (ms)	Migrated p50, mean $\pm$ 95% CI (ms)	Monolith p95, mean $\pm$ 95% CI (ms)	Migrated p95, mean $\pm$ 95% CI (ms)
Category	1.1946 $\pm$ 0.5366	1.3608 $\pm$ 0.7943	2.0680 $\pm$ 0.6595	2.1741 $\pm$ 1.1476
Craft list	95.7724 $\pm$ 4.6287	71.3246 $\pm$ 13.0936	109.8176 $\pm$ 7.4894	80.9153 $\pm$ 22.5234
Craft detail (ID 67)	5.7608 $\pm$ 0.3663	4.3393 $\pm$ 1.0157	6.5849 $\pm$ 1.9891	4.9279 $\pm$ 1.2825

Source: repeated runtime measurement dataset. Each value is the mean of three run-level statistics; the interval is the two-sided 95% confidence-interval half-width across the three runs.

Category latency was comparable across the two conditions, whereas Craft list and Craft detail showed lower descriptive p95 values in the migrated condition. The confidence intervals, particularly for Craft list, show measurable run-to-run variation. These results are descriptive local observations rather than production acceptance criteria: no SLA, user-facing latency target, or validated non-inferiority margin was available.

3.11 Architectural Metrics

The Structural Coupling Ratio (M03) fell from 1.000000 to 0 because the extracted Catalog Service does not import business packages from the remaining monolith under the frozen static-call definition. This result must not be interpreted as the elimination of all coupling. The physical database remains shared and the schema was not separated. The frozen C03 boundary analysis recorded M08=0.444444 for foreign-key boundary coupling, while the shared schema still contains the same foreign-key relationships after extraction. Catalog also continues to read Profile, Promotion, and Review data directly from the shared database. In addition, Ordering, cancellation, and expiry paths retain direct writes to craft_size. Therefore, M03=0 demonstrates removal of remaining-monolith source-code imports only; it does not demonstrate database independence, exclusive Stock ownership, or the absence of cross-boundary data coupling.

Structural Independence (M12) increased from 0.823529 to 0.852941 because one additional operation no longer has external dependencies according to the manifest. The number of distinct external targets fell from five to three. M09 Data Ownership Clarity remains 1.000000 based on the five tables with Catalog as logical owner, but Ordering and expiry still write directly to craft_size. All 55 observed public Catalog writes did pass through the service; this finding is not the same as system-wide exclusive Stock ownership.

At the process level, deployment independence changed from NO to YES: Catalog can be stopped and restarted without restarting the monolith or the gateway, then passes the health and Craft-list probes again. Database independence remains NO because both processes use the same physical database. These results therefore demonstrate process-level deployment independence in the experimental environment, not physical data independence.

The main architectural impacts and their interpretation limits are summarized in Table 5.

Table 5. Architectural metrics and trade-offs

Measure	Before	After	Interpretation
M02 External Dependency Ratio	0.833333	0.750000	Decreased; better metric direction
M03 Structural Coupling Ratio	1.000000	0.000000	Decreased on the static-call graph; shared data remains
M12 Structural Independence	0.823529	0.852941	Increased on the observed operations
M09 Data Ownership Clarity	1.000000	1.000000	Unchanged under the rubric; external Stock writer remains
Deployment independence	NO	YES	YES at the process level
Database independence	NO	NO	Unchanged; not yet achieved

Source: final experiment before–after architectural comparison.

A static three-configuration comparison separated the extraction effect from the research-adapter effect. The monolith with direct Notification dependency (A) and a counterfactual extracted Catalog retaining that dependency (B) have identical M02, M03, and M12 values (0.833333, 1.000000, and 0.823529, respectively). Only after applying the research no-op adapter (C) do the values change to 0.750000, 0.000000, and 0.852941. Thus, the observed improvement in these three metrics is attributable to containing the Notification/Firebase dependency through the research adapter under the frozen static dependency definitions, rather than to service extraction alone. This is a static dependency-graph comparison and does not demonstrate production Notification or Firebase delivery.

3.12 Trade-offs and Comparison with Related Work

The transition preserves public URLs through the gateway allowlist and permits process-level rollback, but adds a network/process hop and a new service-availability boundary. Shared database and media storage, direct Stock writers outside Catalog, and local descriptive latency results limit physical autonomy. Therefore, the implementation demonstrates one controlled Strangler transition rather than a complete decomposition of FemaleFit.

Table 6. Comparison of the proposed framework with related studies

Study	Main evidence/input	Decision mechanism	Implementation or evaluation
Cromlech [7]	Operations, entities, cohesion, communication, and replication	Optimization of decomposition quality	Decomposition-oriented evaluation
Kinoshita and Kanuka [8]	Structural and multi-objective decomposition criteria	Multi-objective optimization	Comparative optimization result
Oumoussa and Saidi [9]	Business-process models and semantic information	NLP, semantic analysis, and clustering	Candidate identification from business artifacts
Maharjan et al. [10]	Static dependency graph and source-code features	Variational autoencoder and C-means clustering	Food-to-Go benchmark
Yilmaz and Buzluca [11]	Microservice quality attributes	Fuzzy-logic quality model	Candidate-quality assessment
Proposed FemaleFit framework	Business capability, source code, database evidence, M01–M12, migration risk	DDD mapping → metrics → W1–W3 sensitivity → practical top set → qualitative decision	One C03 Strangler extraction; route reconciliation; functional, runtime, and architectural evaluation

The proposed framework does not claim to outperform the compared methods because no direct benchmark was conducted on the same dataset. Its contribution is the integration of DDD reasoning, source-code and database traceability, quantitative comparison, sensitivity analysis, explicit migration-risk interpretation, and implementation-level evaluation for one documented boundary decision. The resulting evidence remains bounded by the FemaleFit case, the shared-database transition, and the local evaluation environment.

3.13 Limitations

This research is a single-case study of FemaleFit; generalization to other domains, programming languages, team scales, or production topologies must be done analytically and retested. The researcher also acted as the backend developer and system informant. One external reviewer assessed the frozen mapping, but one review does not eliminate interpretation bias. The predefined formulas, Evidence IDs, alternative boundaries, and reproducibility audit provide additional controls.

The framework is a protocol-specific operationalization of extractability; the seven-candidate case does not support statistical factor validation. Construct-related evidence is limited to metric traceability, the frozen scoring rubric, alternative-boundary analysis, weight-sensitivity analysis, and the external review of the frozen 32-record domain mapping. These controls do not constitute broad psychometric validation or eliminate researcher interpretation bias.

No production SLA or validated latency non-inferiority margin was available. The p95 changes are therefore descriptive results from the controlled local workload and must not be interpreted as production performance acceptance criteria.

The Git history before the experiment was insufficient to compute change frequency or co-change, so both constructs were excluded rather than estimated. Runtime evaluation was performed on a single machine over local loopback with sequential requests. Although three independent runs and confidence intervals were added, the evaluation does not assess production concurrency, autoscaling, distributed networks, or resource contention. The implementation scope remains limited to C03. Functional evaluation covers one representative case for each of the 34 operations, including negative/error cases, but does not cover every valid payload or external-service variation. The shared database and media are still used, Stock ownership is not yet exclusive, and the Notification/Firebase adapter affects C03 runtime dependencies without representing production notification delivery. CPU and memory utilization were not measured.

4. CONCLUSION

This study constructs a service boundary decision by connecting domain reasoning and quantitative evidence. Mapping business objectives, capabilities, processes, endpoints, controllers/functions, models, tables, and dependencies produced 13 initial contexts and seven atomic candidates. The candidates were compared using M01–M13, W1–W3, a practical tie, and alternative-boundary analysis. In this framework, explainable means that the basis of the decision can be traced from business evidence to migration risk and evaluation results; the term does not refer to Explainable AI. C03 Craft Catalog was selected from the practical top set because its domain evidence is more concentrated, its core flow had been verified, its route seam can be formulated explicitly, and its migration blast radius was judged more manageable than C01 Identity & Access. C03 was then extracted as a Go/Fiber process behind a Strangler Gateway. A total of 34 unique method–path pairs are routed through the allowlist, while other routes remain handled by the monolith and both processes still use a shared database. In the test environment, the 34 evaluated Catalog contract cases produced the same HTTP status, content type, and response body as the monolith under the defined test inputs. All 540 retained runtime requests completed without unexpected HTTP 5xx across three independent runs per condition. Category latency was comparable across conditions, whereas Craft list and Craft detail showed lower descriptive p95 values after migration. M02 fell from 0.833333 to 0.750000, M03 from 1 to 0 on the static-call graph, and M12 rose from 0.823529 to 0.852941. Catalog can also be restarted separately. These changes indicate improvement in the defined structural metrics and process-level deployment independence, not database independence or a general production-performance gain. The findings are bounded by one case, one service, local sequential requests, three independent local runs per condition, the researcher's dual role, the shared database, and the absence of CPU/memory measurement. The Notification/Firebase adapter also affects the runtime dependencies and does not represent production notification delivery. Future research should test the framework on multiple cases with independent reviewers, concurrent and distributed workloads, database separation, additional boundary extractions, and evaluation of maintainability and operational cost over a longer period.

REFERENCES

- [1] P. I. Habib, A. S. Murdha, H. Agus, and Suhardi, "Architecture Migration From Monolithic to Microservices: Developing Readiness Criteria," *IEEE Access*, vol. 12, pp. 194630–194645, 2024, doi: 10.1109/ACCESS.2024.3504848.
- [2] H. Michael Ayas, P. Leitner, and R. Hebig, "An empirical study of the systemic and technical migration towards microservices," *Empir. Softw. Eng.*, vol. 28, no. 4, Jul. 2023, doi: 10.1007/s10664-023-10308-9.
- [3] I. Oumoussa and R. Saidi, "Evolution of Microservices Identification in Monolith Decomposition: A Systematic Review," *IEEE Access*, vol. 12, pp. 23389–23405, 2024, doi: 10.1109/ACCESS.2024.3365079.
- [4] Vlad Khononov, *Learning Domain-Driven Design: Aligning Software Architecture and Business Strategy*, 1st ed. O'Reilly Media, 2021.

- [5] O. Özkan, Ö. Babur, and M. van den Brand, “Refactoring with domain-driven design in an industrial context: An action research report,” *Empir. Softw. Eng.*, vol. 28, no. 4, Jul. 2023, doi: 10.1007/s10664-023-10310-1.
- [6] O. Al-Debagy and P. Martinek, “Dependencies-based microservices decomposition method,” *International Journal of Computers and Applications*, vol. 44, no. 9, pp. 814–821, 2022, doi: 10.1080/1206212X.2021.1915444.
- [7] G. Quattrocchi, D. Cocco, S. Staffa, A. Margara, and G. Cugola, “Cromlech: Semi-Automated Monolith Decomposition into Microservices,” *IEEE Trans. Serv. Comput.*, vol. 17, no. 2, pp. 466–481, Mar. 2024, doi: 10.1109/TSC.2024.3354457.
- [8] T. Kinoshita and H. Kanuka, “Enhancing Automated Microservice Decomposition via Multi-Objective Optimization,” *IEEE Access*, vol. 12, pp. 55697–55710, 2024, doi: 10.1109/ACCESS.2024.3389700.
- [9] I. Oumoussa and R. Saidi, “Automated Microservices Identification Through Business Process Analysis: A Semantic-Driven Clustering Approach,” *IEEE Access*, vol. 13, pp. 89926–89945, 2025, doi: 10.1109/ACCESS.2025.3571809.
- [10] R. Maharjan, K. Sooksatra, T. Cerny, Y. Rajbhandari, and S. Shrestha, “A Case Study on Monolith to Microservices Decomposition with Variational Autoencoder-Based Graph Neural Network,” *Future Internet*, vol. 17, no. 7, Jul. 2025, doi: 10.3390/fi17070303.
- [11] R. Yılmaz and F. Buzluca, “A fuzzy logic-based quality model for identifying microservices with low maintainability,” *Journal of Systems and Software*, vol. 216, Oct. 2024, doi: 10.1016/j.jss.2024.112143.
- [12] G. Buchgeher, S. Schoberl, V. Geist, B. Dörninger, P. Haindl, and R. Weinreich, “Using Architecture Decision Records in Open Source Projects - An MSR Study on GitHub,” *IEEE Access*, vol. 11, pp. 63725–63740, 2023, doi: 10.1109/ACCESS.2023.3287654.
- [13] Sam Newman, *Monolith to Microservices: Evolutionary Patterns to Transform Your Monolith*. Sebastopol, CA, USA: O’Reilly Media, 2019.
- [14] H. Hassan, M. A. Abdel-Fattah, and W. Mohamed, “A Pattern-Based Framework for Automated Migration of Monolithic Applications to Microservices,” *Big Data and Cognitive Computing*, vol. 9, no. 10, Oct. 2025, doi: 10.3390/bdcc9100253.
- [15] E. T. Nordli, S. G. Haugeland, P. H. Nguyen, H. Song, and F. Chauvel, “Migrating monoliths to cloud-native microservices for customizable SaaS,” *Inf. Softw. Technol.*, vol. 160, Aug. 2023, doi: 10.1016/j.infsof.2023.107230.
- [16] J. Kazanavičius, D. Mažeika, and D. Kalibatiene, “An Approach to Migrate a Monolith Database into Multi-Model Polyglot Persistence Based on Microservice Architecture: A Case Study for Mainframe Database,” *Applied Sciences (Switzerland)*, vol. 12, no. 12, Jun. 2022, doi: 10.3390/app12126189.
- [17] X. Li, D. Amoroso D’aragona, T. Cerny, V. Lenarduzzi, D. Taibi, and A. Janes, “Exploring Microservice Ownership and Organizational Coupling in Open-Source Projects: An Empirical Study,” *Computing*, vol. 107, no. 4, pp. 1–35, Apr. 2025, doi: 10.1007/s00607-025-01454-7.
- [18] D. Faustino, N. Gonçalves, M. Portela, and A. Rito Silva, “Stepwise migration of a monolith to a microservice architecture: Performance and migration effort evaluation,” *Performance Evaluation*, vol. 164, May 2024, doi: 10.1016/j.peva.2024.102411.
- [19] N. Saarimäki, M. Robredo, V. Lenarduzzi, S. Vegas, N. Juristo, and D. Taibi, “Does microservice adoption impact the velocity? A cohort study,” *Empir. Softw. Eng.*, vol. 30, no. 5, Oct. 2025, doi: 10.1007/s10664-025-10673-7.
- [20] A. TaeiZadeh, Z. Lotfi, and A. J. Ramadhan, “Microservices Boundary Determination Migration in DevOps: a Case Study,” *BIO Web Conf.*, vol. 97, Apr. 2024, doi: 10.1051/bioconf/20249700122.
- [21] Y. Abgaz et al., “Decomposition of Monolith Applications Into Microservices Architectures: A Systematic Review,” *IEEE Transactions on Software Engineering*, vol. 49, no. 8, pp. 4213–4242, Aug. 2023, doi: 10.1109/TSE.2023.3287297.