

From Custom Code Inventory to Clean Core: CORE-Fit—A Portfolio-Level Methodology for Rationalizing ERP Extensions in SAP S/4HANA

Abhinav Jain
Independent Researcher, Pennsylvania, USA

Article Info

Article history:

Received August, 2025
Revised August, 2025
Accepted August, 2025

Keywords:

ABAP Cloud;
Clean Core;
CORE-Fit;
Custom Code;
Design Science;
Enterprise Architecture;
ERP Modernization;
Extensibility;
RICEF Rationalization;
SAP BTP;
SAP S/4HANA

ABSTRACT

Enterprise resource planning (ERP) modernization increasingly requires organizations to reconcile two competing objectives: preserving business-specific capabilities and reducing the lifecycle burden created by tightly coupled custom code. Clean Core addresses this tension through fit-to-standard processes, governed extensibility, stable interfaces, and deliberate placement of capabilities inside or outside the transactional ERP core. This paper introduces CORE-Fit (Customization Optimization, Rationalization, and Extensibility Fit), a design-science methodology proposed in this study for rationalizing inherited ERP customizations into five disposition paths: retire or standardize, key-user extension, on-stack developer extension, side-by-side extension, or governed exception and remediation. The underlying principles and related SAP approaches to custom-code rationalization and extension design were already established; CORE-Fit does not claim novelty for retirement, fit-to-standard, or extension technology selection as individual concepts. Its contribution is the specific six-stage portfolio-level decision artifact that integrates business-need validation, standard-fit analysis, transaction and data proximity, lifecycle independence, integration and economic/operational fit, and governance into five explicit disposition outcomes. CORE-Fit is evaluated using an anonymized customer-system assessment of 121 RICEF objects. The resulting disposition comprised 22 objects (18.2%) retired or standardized, 11 (9.1%) classified as key-user extensions, 73 (60.3%) as on-stack developer extensions, 15 (12.4%) as side-by-side extensions, and none as governed exceptions. The results illustrate that Clean Core rationalization need not equate to aggressive externalization: in this portfolio, most surviving requirements remained close to the transactional core through governed developer extensibility. The study presents a case-evaluated methodology for applying these criteria consistently across a legacy customization portfolio.

This is an open access article under the [CC BY-SA](#) license.



Corresponding Author:

Name: Abhinav Jain
Institution: Independent Researcher, Pennsylvania, USA
Email: Abhi105@gmail.com

1. INTRODUCTION

Enterprise resource planning systems are designed to standardize and integrate critical business processes, yet most large implementations accumulate some degree of customization over time. Earlier ERP research consistently describes customization as a trade-off: it can close functional gaps and preserve differentiating processes, but it can also increase maintenance effort, implementation complexity, and the cost of future upgrades [1], [2], [3], [4], [5], [6]. The modernization problem is therefore not simply whether custom development should exist. The more difficult question is how an organization should decide which business capabilities genuinely require extension, which historical customizations can be eliminated, and where the remaining extensions should be placed in a modern architecture.

This problem has become more important as ERP landscapes shift from heavily modified on-premises systems toward cloud-oriented and upgrade-stable architectures. Prior research documents a movement away from direct code modification toward configuration, interfaces, integration, and other forms of extensibility [5], [6]. In the SAP ecosystem, the Clean Core strategy formalizes a similar direction: the transactional core should remain stable and upgrade-ready, while extensions are implemented through supported on-stack mechanisms or loosely coupled side-by-side services depending on the business and technical requirement [7], [8], [9], [10].

The industrial challenge is especially visible during SAP S/4HANA transformations. A legacy environment may contain hundreds or thousands of reports, interfaces, conversions, enhancements, forms, workflows, and other custom developments. Carrying these objects forward on a one-to-one basis reproduces historical technical debt in the target platform. Conversely, eliminating customization indiscriminately can remove legitimate business differentiation,

regulatory controls, or industry-specific capabilities. The transformation therefore requires a repeatable decision process that separates the business requirement from its historical technical implementation.

This paper addresses that problem by introducing CORE-Fit (Customization Optimization, Rationalization, and Extensibility Fit), a portfolio-level methodology developed in this study for the systematic disposition of existing ERP customizations. The name reflects its purpose: optimizing the inherited customization portfolio, rationalizing whether each requirement should continue to exist, and then selecting the best-fit extensibility model for the surviving requirement. CORE-Fit is intended for existing custom-object portfolios before or during an S/4HANA transformation. Its first objective is disposition: whether a legacy capability should be retired, standardized, redesigned on-stack, externalized side-by-side, or temporarily managed as an exception. Three research questions guide the study:

- a. RQ1: Which business and technical criteria are most useful for separating enduring business requirements from legacy ERP implementations?
- b. RQ2: How can those criteria be organized into a repeatable portfolio-level decision framework for Clean Core transformation?
- c. RQ3: How can the framework map surviving requirements to modern extensibility patterns without forcing either unnecessary externalization or unnecessary coupling to the ERP core?

The individual Clean Core principles incorporated within CORE-Fit are not claimed as new. Standard adoption, key-user extensibility, developer extensibility, released interfaces, side-by-side architecture, and exception governance are established in ERP literature and SAP guidance [5], [6], [8], [10], [11]. Relevant SAP prior art also addresses custom-code transparency and rationalization, extension-use-case

assessment, and extension-technology selection [9], [12], [13], [14]. CORE-Fit therefore does not claim novelty for custom-code retirement, fit-to-standard, or extension selection in isolation. Its contribution is a six-stage portfolio-level artifact that combines business necessity, standard fit, transaction and data proximity, change velocity and lifecycle independence, integration plus economic/operational fit, and governance into five explicit disposition outcomes. The methodology is grounded in design-science principles [15] and evaluated against an anonymized customer-system assessment of 121 RICEF objects.

2. BACKGROUND AND RELATED WORK

2.1 ERP Customization and Lifecycle Cost

ERP customization has long been recognized as neither universally desirable nor universally avoidable. Light [1] showed through case studies that customization can deliver organizational benefits while also creating implications for future maintenance. Brehm, Heinzl, and Markus [2] expanded the discussion by describing a spectrum of ERP tailoring choices rather than a simple standard-versus-custom dichotomy. Their work is particularly relevant to Clean Core thinking because it highlights that configuration, bolt-ons, interfaces, user exits, and direct modifications impose different maintenance and upgrade consequences.

Subsequent empirical work reinforced the lifecycle dimension. Koch and Mitteregger [3] found a statistically significant relationship between greater ERP customization and higher support effort. Dittrich and Vaucouleur [4] similarly examined the development and upgrade practices surrounding customized standard systems, showing that customization creates software-engineering challenges distinct from greenfield application development. These studies suggest that architectural decisions made during implementation

continue to influence operational cost well after go-live.

2.2 Decision Frameworks for ERP Customization

A second stream of research focuses on improving customization decisions rather than merely documenting their consequences. Parthasarathy and Daneva [5] proposed a requirements-based customization framework and used the Analytic Hierarchy Process to prioritize feasible alternatives. Hustad and Stensholt [6] later developed a decision-support framework based on ERP-expert interviews, emphasizing that organizations must weigh drivers, benefits, and consequences when deciding whether to customize. These studies establish that customization can be treated as a structured decision problem rather than an ad hoc technical choice.

2.3 Shift Toward Cloud-Oriented Extensibility

The technical meaning of ERP customization is also changing. Hansen, Haddara, and Langseth [16] observed that cloud ERP reduces or changes traditional customization possibilities and makes the upgrade implications of customization more prominent. Abendroth, Bender, and Gronau [7], based on a systematic review of technical customization definitions, identify a broader evolution from direct modification toward interface- and integration-oriented customization in cloud ERP environments. This evolution provides the academic context for Clean Core: business differentiation still exists, but the technical mechanisms used to realize it are becoming more decoupled and lifecycle-aware.

2.4 Clean Core and SAP Extensibility

SAP describes Clean Core as an approach for keeping ERP environments agile and upgrade-ready by reducing unnecessary complexity and governing business processes, extensibility, data, integration, and operations [11]. For

extensibility specifically, SAP guidance distinguishes on-stack approaches—including key-user and developer extensibility—from side-by-side extensions on SAP Business Technology Platform (BTP) [8], [10]. ABAP Cloud provides released interfaces and language restrictions intended to support lifecycle-stable on-stack development, while side-by-side architecture enables loosely coupled applications to evolve outside the S/4HANA lifecycle [8], [10].

The transformation problem addressed in this study begins before a target extension mechanism is selected. A legacy ERP portfolio contains technical objects whose underlying business requirements may have changed, become redundant, or become supportable through standard functionality. CORE-Fit therefore starts with portfolio rationalization: it first determines whether the business requirement should survive and only then determines the appropriate architectural disposition for the surviving requirement.

2.5 Existing SAP Approaches to Custom-Code Rationalization and Extension Decisions

Relevant SAP prior art covers complementary parts of the problem addressed by CORE-Fit. Intelligent Custom Code Management (iCCM) provides transparency into existing custom code and identifies candidates for retirement, redesign, and replatforming [12]. The SAP Application Extension Methodology (AEM), available since 2021 and at version 2.3 by March 2025, structures extension-use-case assessment, extension-technology assessment, and target-solution definition [13]. The Extension Architecture Guide, publicly documented by late 2023, provides decision guidance for selecting SAP extension technologies and is positioned within the same SAP BTP Guidance Framework [14]. SAP Clean Core

guidance also describes a transformation sequence that includes analysis of existing custom code, retirement of unused objects, creation of redesign/replatform lists, adoption of extension guidelines, and governance [9]. These approaches establish that custom-code rationalization and guided extension selection were already recognized practices before this study.

2.6 Research Gap

Against this prior art, CORE-Fit makes a deliberate portfolio driven contribution. It does not claim invention of custom-code retirement, fit-to-standard analysis, on-stack versus side-by-side extensibility, or extension-technology decision guidance. Instead, CORE-Fit contributes a specific six-stage portfolio-level decision artifact that integrates: (1) business necessity, (2) standard fit, (3) transaction and data proximity, (4) change velocity and lifecycle independence, (5) integration plus economic and operational fit, and (6) placement governance. These criteria are resolved into five explicit disposition outcomes—retire/standardize, key-user extension, on-stack developer extension, side-by-side extension, or governed exception. The research contribution is therefore the synthesis and operationalization of these criteria in one repeatable portfolio-disposition model, together with its evaluation on a 121-object industrial case.

3. RESEARCH METHOD

3.1 Design-Science Orientation

The study follows a design-science orientation because its primary objective is to create and evaluate a practical artifact—a decision methodology—that improves an organizational problem-solving capability. [15] characterize design science as the creation and evaluation of artifacts that extend human and organizational capabilities. CORE-Fit structures the reasoning required to transform an inherited custom-code

inventory into a rationalized target extensibility portfolio.

CORE-Fit was developed by triangulating three knowledge sources: (1) peer-reviewed ERP customization research, addressing maintenance, upgrade risk, and customization decisions; (2) contemporary SAP Clean Core, custom-code-management, and extensibility guidance, including the prior approaches discussed in Section II-E; and (3) practitioner evidence from an anonymized S/4HANA transformation in which 121 RICEF objects were evaluated in the customer system rather than automatically carried forward. In light of this prior art, the proposed novelty is intentionally limited to the integration of these inputs into the CORE-Fit six-stage decision sequence, its explicit economic and operational-fit criterion, and its portfolio-level application evaluated in this study.

3.2 Industrial Case Context

The case context is an anonymized SAP S/4HANA transformation involving complex, industry-specific processes and a mixed portfolio of reports, interfaces, conversions, enhancements, forms, workflows, and custom applications. The transformation introduced modern S/4HANA development and integration patterns including CDS, the ABAP RESTful Application Programming Model (RAP), OData, SAP Fiori, ABAP Cloud, APIs, event-driven integration, and SAP BTP services. A technical assessment of 121 RICEF objects examined whether each capability should be retired or standardized, implemented through key-user extensibility, redesigned as an on-stack developer extension, implemented side-by-side, or retained temporarily as a governed exception.

The assessment experience produced a recurring observation: object-level discussions became more productive when teams first articulated the business requirement and only then

examined the technical implementation. This insight became the central design principle of CORE-Fit. The framework does not assume that all custom code is harmful, nor does it assume that all innovation belongs outside the ERP system. Instead, it seeks a fit-for-purpose placement that preserves transactional integrity, performance, security, maintainability, and business differentiation while accounting for lifecycle and operating cost.

3.3 Evaluation Logic

Evaluation combines analytical scenario testing with descriptive evidence from the 121-object customer-system assessment. The framework is examined against representative extension scenarios—redundant legacy functionality, transaction-local validation, independent business applications, workflow orchestration, integration-heavy requirements, high-volume data or document workloads, and legacy enhancements that rely on unstable internals. The empirical disposition provides an additional portfolio-level check: 22 of the assessed RICEF objects (18.2%) were retired or standardized, 11 (9.1%) were classified as key-user extensions, 73 (60.3%) as on-stack developer extensions, and 15 (12.4%) as side-by-side extensions; no objects in this assessment were ultimately classified as governed exceptions. These results are descriptive of one transformation context rather than evidence of a universal target distribution.

4. CORE-FIT FRAMEWORK FOR CLEAN CORE RATIONALIZATION

CORE-Fit is a six-stage decision process. The sequence is intentional: business necessity and standard fit are evaluated before technical placement, while cost and operational fit are evaluated before a final target is approved. This avoids a common modernization anti-pattern in which teams

begin by asking how or where to rebuild an existing object rather than whether the object

should continue to exist and what placement is sustainable over its lifecycle.

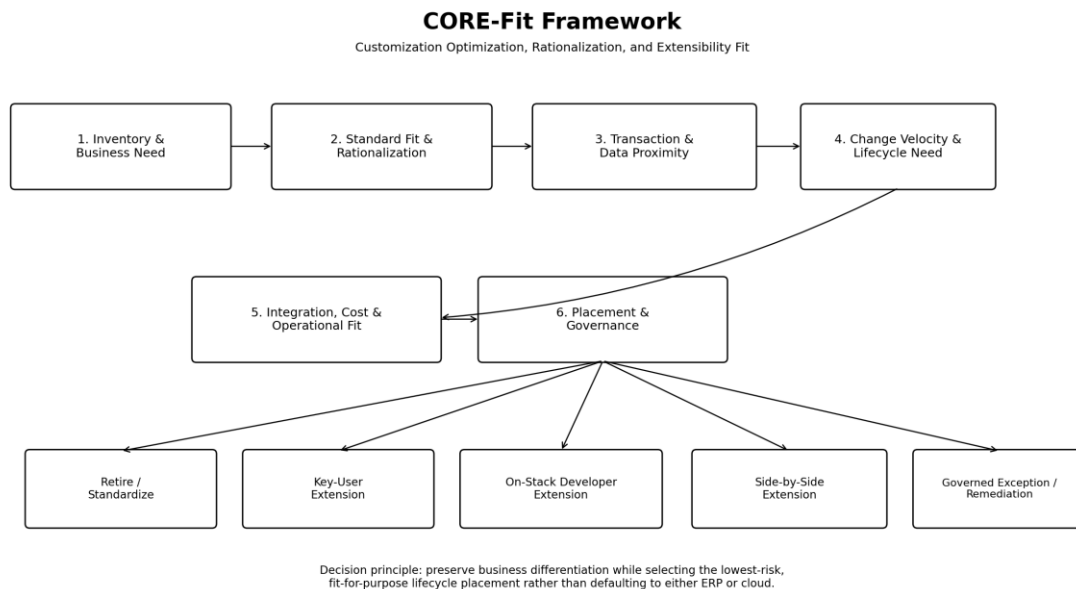


Figure 1. CORE-Fit Framework for Clean Core rationalization and Extensibility Placement

4.1 Stage 1 – Inventory and Business Necessity

The first stage establishes what the object actually does, who uses it, which process it supports, whether the requirement is regulatory or differentiating, and whether the capability is still active. Objects that lack a current business owner, have no usage, duplicate another solution, or support an obsolete process become candidates for retirement before any technical migration work is considered. This stage intentionally decouples the business requirement from the legacy object name, program type, or implementation history.

4.2 Stage 2 – Standard Fit and Rationalization

If the requirement remains valid, the next question is whether the target S/4HANA release can satisfy it through standard functionality or configuration. A strong standard fit favors adoption of the standard process and retirement of the custom implementation. Partial fit may require a small key-user or developer extension rather than full custom redevelopment. The objective is not standardization for

its own sake; it is the removal of technical uniqueness that no longer provides business value.

4.3 Stage 3 – Transaction and Data Proximity

Requirements that must execute inside the same logical unit of work as an SAP business transaction, enforce synchronous validation, update standard and custom data atomically, or perform data-intensive operations close to S/4HANA have a stronger on-stack affinity. SAP Clean Core and ABAP Cloud guidance identify transactional consistency, proximity to SAP data, and lifecycle-stable extension mechanisms as important architectural considerations [8], [10], [11]. In a Clean Core architecture, such extensions should favor released APIs, released extension points, ABAP Cloud, CDS, RAP, and other lifecycle-stable mechanisms where available.

4.4 Stage 4 – Change Velocity and Lifecycle Independence

Capabilities that change frequently, serve a distinct user population, combine multiple systems, experiment with new technology, or require an independent release cadence

have a stronger side-by-side affinity. Decoupling is particularly valuable when the extension does not need to share the S/4HANA transactional lifecycle. Typical examples include custom experience applications, cross-system orchestration, AI-enabled functions, and multi-system applications implemented on SAP BTP.

4.5 Stage 5 – Integration, Cost, and Operational Fit

The framework next evaluates how many systems participate in the process, whether communication is synchronous or asynchronous, how much data crosses the boundary, what retention and retrieval patterns apply, and whether an event-driven model is appropriate. An integration-heavy requirement should not automatically become an ERP customization; APIs, SAP Integration Suite, and event-based patterns can create explicit boundaries. Equally, high data or document volume does not by itself justify moving storage or processing to BTP or another cloud service. External cloud persistence can introduce recurring storage, data-

transfer, backup, retention, and operational costs. **CORE-Fit** therefore treats placement as an economic as well as architectural decision. Transactional proximity, performance, compliance, existing enterprise repositories, lifecycle ownership, and total cost should be considered together. A supported on-stack or existing enterprise storage pattern may be preferable when it satisfies Clean Core boundaries more economically; side-by-side storage or processing is selected when its lifecycle independence or integration value justifies the additional platform cost.

4.6 Stage 6 – Placement and Governance

The final stage records a target disposition and the reasoning behind it. A decision is considered complete only when the business owner, architectural rationale, target technology class, interface contract, security responsibility, test impact, and exception status are documented. This converts Clean Core from a one-time transformation principle into an operational governance mechanism.

Table 1. CORE-Fit Decision Dimensions

Decision dimension	Key question	Architectural implication
Business necessity	Is the requirement still used, mandated, differentiating, or measurable in value?	Low necessity favors retirement; high necessity proceeds to fit analysis.
Standard solution fit	Can standard S/4HANA functionality or configuration meet the requirement?	High fit favors standardize/retire custom implementation.
Transactional proximity	Must logic participate in the same SAP transaction or LUW?	High proximity favors governed on-stack extension.
Data proximity / volume / economics	Where can the data be processed or retained with the required transactional consistency, performance, compliance, and acceptable lifecycle cost?	High transaction/data proximity may favor on-stack processing. Side-by-side placement should be justified by lifecycle or integration value after storage, transfer, retention, and operating cost are considered.
Change velocity	How frequently must the capability evolve independently of ERP releases?	High velocity favors side-by-side lifecycle independence.
Integration breadth	How many systems, APIs, events, or external data sources participate?	Broad integration favors explicit integration layer and/or side-by-side orchestration.
Upgrade risk	Does the solution depend on	High risk triggers redesign or

Decision dimension	Key question	Architectural implication
	unreleased internals, modifications, or fragile enhancement techniques?	governed exception with remediation plan.

Table 2. CORE-Fit Target Dispositions

Disposition	When appropriate	Typical target approach
Retire / standardize	Requirement obsolete, duplicated, or adequately met by standard capability.	Remove legacy object; adopt standard process/configuration.
Key-user extension	Small field, UI, form, rule, or low-code adaptation within supported boundaries.	Use in-app/key-user extensibility when sufficient.
On-stack developer extension	Tight transaction or data proximity is required and a released extension model exists.	Use ABAP Cloud, RAP/CDS, released APIs and extension points.
Side-by-side extension	Independent lifecycle, broad integration, or non-ERP capabilities dominate and the decoupling benefit justifies the additional platform footprint.	Use SAP BTP capabilities such as CAP, workflow/automation, APIs/events, and data services where appropriate; avoid externalization based on volume alone.
Governed exception	Business-critical requirement cannot yet be implemented through preferred clean mechanisms.	Document exception, owner, risk, upgrade checks, and time-bound remediation path.

5. CASE-BASED APPLICATION

5.1 Legacy Portfolio Rationalization

The S/4HANA transformation context that informed the framework contained 121 RICEF objects assessed directly in the customer system. The assessment did not treat the legacy landscape as a migration checklist. Instead, each object was examined against the transformed business process and the target architecture. The final disposition comprised 22 objects (18.2%) retired or standardized, 11 (9.1%) classified as key-user extensions, 73 (60.3%) as on-stack developer extensions, 15 (12.4%) as side-by-side extensions, and none as governed exceptions. The exact object counts and corresponding percentages summarize the distribution of the assessed portfolio without disclosing customer identity or business-sensitive details.

The most useful assessment pattern was requirement-first rather than code-first. Teams asked whether the business need remained valid, whether S/4HANA standard functionality had changed, and what degree of transactional proximity the target requirement actually required. Only after those questions were answered did the assessment consider RAP, CDS, OData, Fiori, ABAP Cloud, BTP services, APIs, or event-driven integration as implementation choices. The distribution is noteworthy because it did not produce a cloud-heavy result: 73 of 121 objects (60.3%) remained on-stack through governed developer extensibility, while 15 (12.4%) required side-by-side placement and 11 (9.1%) were addressed through key-user extensibility.

5.2 Portfolio Disposition Results

Table 3. Disposition of the 121 Anonymized RICEF Objects Assessed in the Customer System

Disposition	Object count	Share of 121 objects
Retire / standardize	22	18.2%
Key-user extension	11	9.1%
On-stack developer extension	73	60.3%

Disposition	Object count	Share of 121 objects
Side-by-side extension	15	12.4%
Governed exception	0	0.0%

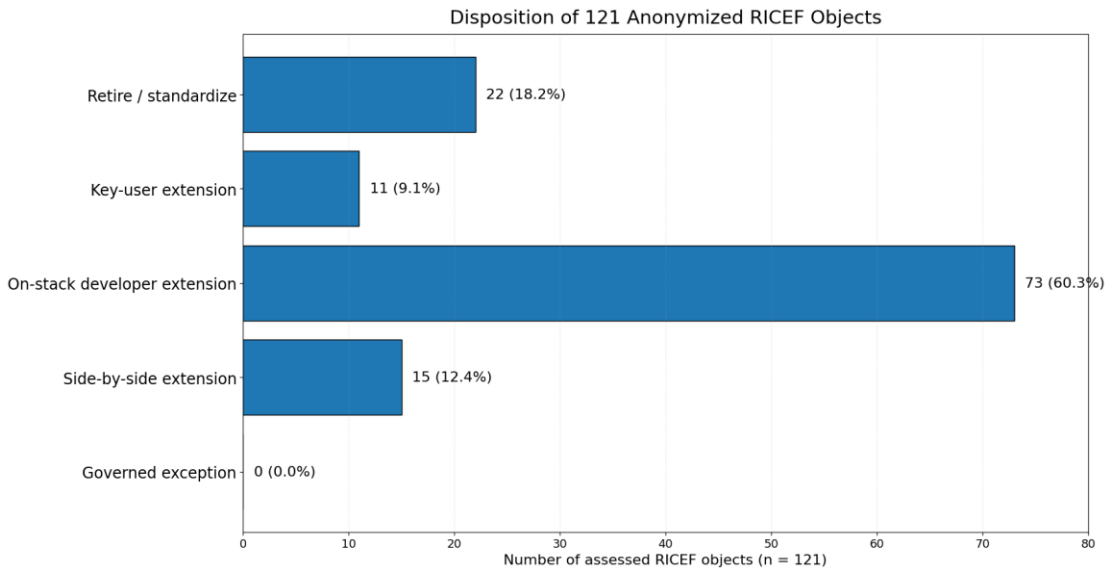


Figure 2. Portfolio-Level Disposition Results from the 121-object Anonymized RICEF Assessment

The distribution should be interpreted as an outcome of the assessed requirements rather than a prescribed Clean Core target. In particular, the 11 objects (9.1%) key-user result shows that a small subset of requirements could be addressed through supported in-app adaptation without requiring developer extensibility. Likewise, the absence of governed exceptions indicates that the assessed requirements could be mapped to standard or supported extension patterns without retaining a formally accepted non-clean implementation.

5.3 Representative Disposition Scenarios

Table 4. Representative Case-Based Applications of CORE-Fit

Scenario	Dominant CORE-Fit signals	Disposition	Rationale
Legacy report duplicates target standard capability	Standard fit is high; no unique processing remains.	Retire / standardize	Avoid redevelopment; validate output and user transition.
Transaction-time validation of business data	Logic must execute synchronously with the ERP transaction.	On-stack developer extension	Use released extension point / ABAP Cloud pattern where available.
Cross-system workflow and approval	Process spans ERP and non-ERP participants and changes independently.	Side-by-side / orchestration	Workflow/automation service with API-based transaction updates.
High-volume documents or data processing	Volume is high, but transaction coupling, retention, retrieval patterns, existing repositories, and platform economics vary.	Context-dependent placement	Compare supported on-stack, enterprise-repository, and side-by-side options using performance, compliance, lifecycle ownership, and total cost. Do not externalize solely because of volume.
Legacy enhancement	Business need remains, but	Redesign or	Move to released

Scenario	Dominant CORE-Fit signals	Disposition	Rationale
uses unstable internal object	implementation creates upgrade risk.	governed exception	API/extension point; track exception until remediated.

5.4 Observed Benefits of the Decision Model

The case-based evaluation indicates four practical benefits. First, the framework reduces “lift-and-shift” bias by making retirement and standardization explicit outcomes; 22 objects (18.2%) of the assessed portfolio fell into this category. Second, it improves architectural consistency by using the same criteria across RICEF categories. Third, the 73-object (60.3%) on-stack result illustrates an important interpretation of Clean Core: it does not mean “move everything to BTP.” Transactional proximity and released extension mechanisms can justify substantial on-stack developer extensibility. Fourth, the framework makes exceptions visible even when none are ultimately required; the evaluated portfolio contained no governed exceptions after assessment.

The framework also changes the governance conversation. Rather than asking only “Can this be built in S/4HANA?” or “Can this be moved to BTP?”, the decision record asks “Where should this capability live over its lifecycle, what coupling is justified by the business requirement, and what is the operational and economic consequence of that placement?” This reframing is consistent with the broader Clean Core objective of maintaining a stable transactional foundation without turning cloud externalization into an objective in itself.

6. DISCUSSION

6.1 Contribution to ERP Customization Research

CORE-Fit contributes to prior customization research not by introducing custom-code retirement or extension selection as new ideas, but by joining several established decision

concerns into one portfolio-disposition artifact. Earlier academic work structures ERP customization decisions [5], [6]; SAP iCCM addresses custom-code transparency and candidates for retirement, redesign, and replatforming [12]; and AEM together with the Extension Architecture Guide provides structured guidance for extension use cases and technology choices [13], [14]. CORE-Fit adds a single repeatable sequence across inherited objects in which continued business necessity, standard fit, transaction/data proximity, lifecycle independence, economics, operations, and governance are evaluated before one of five target dispositions is recorded.

Second, the framework combines lifecycle placement with cost and operational fit. Earlier literature establishes that customization can increase maintenance and support burden [1], [3] while cloud-ERP research documents the shift toward cloud and interface-based customization [7], [16]. CORE-Fit operationalizes that evolution without assuming that decoupling is always economically superior. It asks how much transactional coupling is required, what lifecycle independence is valuable, and whether the additional platform, storage, transfer, and operating footprint of an external solution is justified.

6.2 Portfolio-Level Scope of CORE-Fit

CORE-Fit is designed as a portfolio-level disposition methodology for transformation programs. It does not replace detailed extension-design or technology-selection guidance; its scope is the consistent disposition of an inherited customization portfolio before detailed solution design. It starts with an object inventory, separates requirement retention from implementation retention, records a

target disposition, and applies architectural, economic, operational, and governance criteria before the surviving requirement proceeds into implementation design. This makes the methodology suitable for repeated assessment across a large RICEF portfolio while remaining complementary to more detailed technology-specific guidance.

6.3 *Practical Governance Implications*

For enterprise architecture teams, the most important operational implication is that Clean Core requires a decision record, not merely a technology standard. Governance should capture the business owner, requirement classification, standard-fit assessment, placement rationale, released interface or extension point, integration boundary, expected change cadence, and exception status. This allows architecture reviews to be repeated consistently across waves and prevents different teams from applying incompatible definitions of “clean.”

The framework also supports a measurable portfolio backlog. In the evaluated case, 22 objects (18.2%) of the objects reduced migration scope through retirement or standard adoption, 11 objects (9.1%) became a key-user-extension backlog, 73 objects (60.3%) became an on-stack developer-extension backlog, and 15 objects (12.4%) became a side-by-side backlog. No objects were assigned to governed exception in this specific assessment. This zero-value category should not be interpreted as generally unnecessary; it reflects the characteristics of this portfolio and demonstrates that CORE-Fit does not impose a target percentage for any extensibility model.

6.4 *Limitations and Threats to Validity*

The study has four limitations. First, the framework is grounded primarily in a single SAP S/4HANA transformation and a 121-object RICEF portfolio, which limits generalizability across organizations, industries, and ERP

vendors. Second, the empirical evaluation is descriptive rather than longitudinal: the disposition percentages are reported, but post-implementation measures such as defect rates, upgrade effort, regression-test effort, support cost, and change lead time are not yet available. Third, SAP-specific mappings to ABAP Cloud, RAP, BTP, APIs, and events will evolve as product capabilities change. Fourth, architectural decisions are influenced by non-technical constraints such as skills, licensing, infrastructure and cloud cost, security policy, operating model, and delivery timelines. Future research should evaluate CORE-Fit across multiple transformation programs and test whether its use improves lifecycle outcomes and decision consistency over time.

7. CONCLUSION AND FUTURE WORK

Clean Core is often described as a technology or custom-code objective, but the more fundamental challenge is architectural decision quality. Organizations must decide which inherited capabilities still matter, which can be standardized, and which require extension close to or outside the transactional ERP core. A modern ERP transformation succeeds not when all customization disappears, but when each remaining extension has a defensible business purpose, a supported technical boundary, and a lifecycle model that does not unnecessarily obstruct future change.

This paper proposed CORE-Fit as a staged portfolio-level methodology for rationalizing an inherited ERP customization portfolio in support of Clean Core transformation. The individual concepts it uses—and the broader ideas of custom-code rationalization and guided extension selection—precede this study and are represented in both academic literature and SAP practices such as iCCM, AEM, and extension decision guidance [5], [6], [9], [12], [13], [14]. The contribution of CORE-Fit is

specific: a six-stage portfolio-level decision structure, five explicit disposition outcomes, an explicit economic and operational-fit dimension, and consistent application of those elements to a 121-object industrial portfolio. In the anonymized S/4HANA case, 121 RICEF objects were assessed: 22 objects (18.2%) were retired or standardized, 11 (9.1%) were classified as key-user extensions, 73 (60.3%) as on-stack developer extensions, 15 (12.4%) as side-by-side extensions, and none as governed exceptions. The distribution illustrates that a Clean Core outcome can remain predominantly on-stack when transaction and data proximity justify governed developer extensibility.

Future work should validate CORE-Fit using object-level datasets from multiple transformation programs and through independent classification by enterprise architects. Useful longitudinal measures include upgrade remediation effort, regression-test effort, support incidents, change lead time, extension operating cost, cloud/platform cost, and exception aging. Such evidence would help determine whether the framework improves not only initial architecture decisions but also the long-term maintainability and economics of ERP modernization.

REFERENCES

- [1] B. Light, "The maintenance implications of the customization of ERP software," *J. Softw. Maint. Evol. Res. Pract.*, vol. 13, no. 6, pp. 415–429, 2001, doi: <https://doi.org/10.1002/smr.240>.
- [2] L. Brehm, A. Heinzl, and M. L. Markus, "Tailoring ERP systems: a spectrum of choices and their implications," in *Proceedings of the 34th annual Hawaii international conference on system sciences*, IEEE, 2001, pp. 9-pp. doi: <https://doi.org/10.1109/HICSS.2001.927130>.
- [3] S. Koch and K. Mitteregger, "Linking customisation of ERP systems to support effort: an empirical study," *Enterp. Inf. Syst.*, vol. 10, no. 1, pp. 81–107, Jan. 2016, doi: [10.1080/17517575.2014.917705](https://doi.org/10.1080/17517575.2014.917705).
- [4] Y. Dittrich and S. Vaucoleur, *Customization and Upgrade of ERP systems.: An empirical Perspective*. IT University of Copenhagen, 2008.
- [5] S. Parthasarathy and M. Daneva, "Customer requirements based ERP customization using AHP technique," *Bus. Process Manag. J.*, vol. 20, no. 5, pp. 730–751, 2014, doi: <https://doi.org/10.1108/BPMJ-04-2013-0044>.
- [6] E. Hustad and J. Stensholt, "Customizing ERP-systems: A framework to support the decision-making process," *Procedia Comput. Sci.*, vol. 219, pp. 789–796, 2023, doi: <https://doi.org/10.1016/j.procs.2023.01.352>.
- [7] A. Abendroth, B. Bender, and N. Gronau, "The Evolution of Original ERP Customization: A Systematic Literature Review of Technical Possibilities," *26th Int. Conf. Enterp. Inf. Syst. (ICEIS 2024)*, vol. 1, pp. 17–27, Apr. 2024, doi: [10.5220/0012305500003690](https://doi.org/10.5220/0012305500003690).
- [8] SAP SE, "ABAP Cloud: Background concepts and overview," SAP.
- [9] SAP SE, "Clean Core Roadshow: Your Path to Clean," SAP.
- [10] SAP SE, "Extend SAP S/4HANA in the cloud and on premise with ABAP-based extensions (Version 2.2)," SAP.
- [11] SAP SE, "Clean core extensibility for SAP S/4HANA Cloud (Version 1.0)," SAP.
- [12] SAP SE, "SAP MaxAttention Innovation Workshop 'Intelligent Custom Code Management,'" SAP Community.
- [13] SAP SE, "SAP Application Extension Methodology: Version history (Version 2.3)," SAP.
- [14] M. Matha, "Introducing SAP Application Extension Methodology version 2.2," SAP Community.
- [15] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design science in information systems research1," *MIS Q.*, vol. 28, no. 1, pp. 75–106, 2004, doi: <https://doi.org/10.2307/25148625>.
- [16] H. F. Hansen, M. Haddara, and M. Langseth, "Investigating ERP System Customization: A Focus on Cloud-ERP," *Procedia Comput. Sci.*, vol. 219, pp. 915–923, 2023, doi: <https://doi.org/10.1016/j.procs.2023.01.367>.