Preprints.org

Article

Not peer-reviewed version

BPC4MSA: A Microservices-Based Framework for Business Process Compliance in Cloud Environments

N. Long Ha *, S. Huong Do, Quan Truong Tan, Thomas M. Prinz

Posted Date: 11 December 2025

doi: 10.20944/preprints202512.0916.v1

Keywords: business process compliance; cloud computing; Design Science Research; microservices; regulatory compliance

Preprints.org is a free multidisciplinary platform providing preprint service that is dedicated to making early versions of research outputs permanently available and citable. Preprints posted at Preprints.org appear in Web of Science, Crossref, Google Scholar, Scilit, Europe PMC.

Copyright: This open access article is published under a Creative Commons CC BY 4.0 license, which permit the free download, distribution, and reuse, provided that the author and preprint are cited in any reuse.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 December 2025 doi:10.20944/preprints202512.0916.v1

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

Article

BPC4MSA: A Microservices-Based Framework for Business Process Compliance in Cloud Environments

N. Long Ha [1,*], S. Huong Do [1], Quan Truong Tan [1] and Thomas M. Prinz [2]

[1] University of Economics, Hue University, 99 Ho Dac Di, Hue, 530000, Viet Nam

[2] Course Evaluation Service, Friedrich Schiller University Jena, Am Steiger 3, Haus 1, Jena, 07743, Germany

* Correspondence: hnlong@hueuni.edu.vn

Abstract

The increasing adoption of Microservices Architecture (MSA) and Cloud Computing offers significant agility but introduces substantial challenges for traditional Business Process Compliance (BPC). This paper addresses these challenges using a Design Science Research Methodology (DSRM). We identify persistent research gaps, including a "Broken Feedback Loop" between design-time and runtime systems. To address these gaps, we present BPC4MSA, a cloud-native framework for holistic BPC management spanning the full lifecycle—from design-time verification to resilient, asynchronous runtime auditing—derived from systematic requirements analysis. This work's primary contribution is the BPC4MSA framework, a novel conceptual artifact whose design theoretically addresses these gaps. We evaluate this design via a dual-method approach: first, a Systematic Literature Review confirms the framework's conceptual novelty against the state-of-the-art. Second, a preliminary empirical study of a runtime prototype component is conducted not as a performance benchmark, but as a feasibility check. This check confirms the expected architectural trade-offs of its event-driven pattern, validating its suitability for asynchronous auditing use cases and highlighting critical implementation challenges, such as consumer scaling.

Keywords: business process compliance; cloud computing; Design Science Research; microservices; regulatory compliance

1. Introduction

1.1. Background and Motivation

In contemporary process-oriented business environments, organizations increasingly rely on well-defined business processes to ensure operational efficiency and achieve strategic objectives. The adoption of Business Process Management (BPM) has become imperative for managing organizational complexity and adapting to dynamic market conditions [1]. Alongside operational goals, however, organizations face mounting pressure to ensure their processes comply with a growing number of laws, regulations, and internal policies [2]. Business Process Compliance (BPC), defined as the alignment of business process execution with these prescriptive documents [3,4], is crucial for risk management and maintaining stakeholder trust.

Concurrently, the technological landscape is rapidly evolving with the widespread adoption of Microservices Architecture (MSA) [5] and Cloud Computing. While offering benefits in agility and scalability, these modern environments present significant challenges to traditional BPC approaches. The inherent characteristics of MSA/Cloud—including distribution, dynamism, polyglot persistence, and decentralized control—fundamentally clash with conventional, often centralized, compliance mechanisms. This clash manifests in several critical, unresolved problems:

- Complex Cross-Service Auditing: Reconstructing a complete, reliable audit trail for a business process that traverses multiple, often ephemeral, microservices is non-trivial.

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 December 2025 doi:10.20944/preprints202512.0916.v1

- Slow Policy Adaptation: Traditional methods for updating compliance rules are too slow for the rapid deployment cycles common in MSA environments.
- Impedance Mismatch: Centralized, heavyweight compliance tools create an impedance mismatch with the lightweight, decentralized paradigm of cloud-native microservices.

A systematic review of the state-of-the-art reveals that these issues contribute to persistent and significant research gaps. These include the "Last Mile" Problem of translating ambiguous legal text into executable rules, a "Broken Feedback Loop" between design-time analysis and runtime monitoring, and a prevalent lack of scalable and resilient Automated Verification capabilities in existing solutions. This research confronts the foundational challenge of holistically managing BPC within combined MSA/Cloud environments, a gap explicitly noted in recent literature surveys [6,7]

1.2. Research Objectives & Approach

The primary goal of this research is to design, develop, and evaluate a novel framework, BPC4MSA, specifically engineered for holistic Business Process Compliance (BPC) management, spanning the full lifecycle from design-time verification to resilient, asynchronous runtime auditing within integrated Microservices Architecture (MSA) and Cloud environments. This framework aims to address the problem and gap identified in Section 1.1 by providing capabilities such as scalable compliance checking, runtime monitoring and auditing across distributed services, agile policy management, and seamless integration with cloud-native ecosystems.

To achieve this goal, this research employs a Design Science Research Methodology (DSRM) [8,9], following a structured process adapted from established DSRM guidelines and exemplified in related works. The research followed the following distinct phases:

1. Problem Identification and Motivation (Section 1.1), where we identify the significant challenges traditional BPC approaches face in modern MSA/Cloud settings and motivates the necessity for a new, specialized framework.
2. Definition of Objectives for a Solution: Stemming from the identified problem and the acknowledged literature gap concerning established BPC practices in MSA/Cloud, the precise objectives for the BPC4MSA framework were defined through a systematic requirements derivation process. This process, detailed in Section 3, integrates First-Principles Analysis and Constructed Scenario Analysis [10], resulting in a synthesized set of functional and non-functional requirements that constitute the explicit objectives for the artifact.
3. Design and Development (Section 4), which details the architectural design and component development of the BPC4MSA framework artifact. This artifact was created specifically to instantiate a solution meeting the objectives (requirements) defined in Section 3.
4. Evaluation (Sections 5 & 6), which presents a dual-method evaluation of the artifacts. First, in Section 5, we conduct a Design Evaluation, where the conceptual framework artifact is assessed. Following Polyvyanyy et al. [11], we use a Systematic Literature Review as the method to rigorously validate the framework's conceptual novelty and its completeness in addressing the identified research gaps. Second, in Section 6, we conduct a Prototype Evaluation, where the instantiated prototype artifact is assessed via a preliminary, non-functional empirical study.
5. Communication: This paper serves as the primary vehicle for communicating the entire research process, including the problem context, methodology, artifact design, and findings from both the design and empirical evaluations (DSRM Guideline 7: Communication of Research). Sections 7 (Discussion) and 8 (Conclusion) fulfill this, summarizing findings and outlining future work.

This structured DSRM approach ensures rigor in both the development of the BPC4MSA framework and the evaluation of its contribution to the field.

1.3. Contributions

This research makes the following primary contributions:

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 December 2025 doi:10.20944/preprints202512.0916.v1

We present a systematically derived set of requirements specifically tailored for achieving Business Process Compliance within Microservices, addressing the unique challenges of decentralization and dynamism. Based on these objectives, we detail the conceptual design and description of the BPC4MSA framework artifact, a novel, cloud-native architecture for holistic Business Process Compliance management, spanning from design-time verification to runtime auditing. The conceptual novelty of this design, which theoretically proposes a mechanism to address critical research gaps like the "Broken Feedback Loop", is rigorously established against the state-of-the-art through a systematic literature review (Section 5).

We provide a preliminary feasibility check of the BPC4MSA framework's event-driven runtime component. This check is not a comparative performance benchmark, but a study of the pattern's behavior under load. The results confirm the expected architectural trade-offs of an asynchronous, buffered pattern, demonstrating how it transforms a catastrophic "fast-fail" state into a manageable processing queue. This confirms the pattern's suitability for asynchronous auditing while empirically reinforcing the importance of effective implementation (e.g., consumer scaling) to manage processing latency

1.4. Paper Structure

The remainder of this paper is organized as follows: Section 2 provides necessary background and establishes formal foundations. Section 3 details the requirements derivation for BPC4MSA. Section 4 describes the proposed framework design. Section 5 evaluates the conceptual framework via a Systematic Literature Review. Section 6 details the implementation and preliminary evaluation of a prototype artifact. Section 7 discusses the combined findings, highlighting the architectural trade-offs between efficiency and resilience. Finally, Section 8 concludes the paper.

2. Background and Related Work

This section establishes the foundational concepts and theoretical context for the BPC4MSA framework. It begins by defining the core domains of Business Process Management (BPM), Business Process Compliance (BPC), Microservices Architecture (MSA), and Cloud Technologies. It then presents a structured review of the related academic literature, tracing its evolution to identify the specific research gap this paper addresses. Finally, it establishes the formal foundations used in the framework's design.

2.1. Business Process Management

Business Process Management (BPM) is a discipline that uses the methods, techniques, and software used to define, model, manage, and analyze operational business processes throughout their lifecycle [12,13]. Its goal is to enhance efficiency, productivity, and cost-effectiveness in processes involving people, organizations, and digital systems [1]. A business process consists of structured activities designed to deliver a product or service. Modern BPM must address dynamic and decentralized management needs, such as in Business Process as a Service (BPaaS), where cloud-based platforms enable modeling, customization, and distributed execution [1].

BPM operates through a lifecycle comprising design, execution, and monitoring phases [14]. Compliance is a fundamental aspect of BPM, ensuring adherence to laws, policies, and standards [15]. Business Process Compliance (BPC) occurs across three stages: design-time verification, run-time monitoring, and post-execution auditing [2,14]. Advanced frameworks increasingly employ semantic repositories—integrating business, domain, and regulatory ontologies—to achieve unified and consistent compliance management [14].

2.2. Business Process Compliance

Business Process Compliance (BPC) is the discipline of identifying, implementing, and verifying adherence to business-relevant requirements across the process lifecycle [16]. It ensures that

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 December 2025 doi:10.20944/preprints202512.0916.v1

processes align with laws, standards, and internal policies [17,18]. Noncompliance can lead to financial penalties, litigation, and reputational loss [19,20], underscoring its importance in risk management [21].

Prominent regulatory frameworks such as GDPR, HIPAA, SOX, and PCI DSS govern data protection, financial accountability, and information security [7,22].

BPC operates through design-time, run-time, and post-execution auditing [23], using techniques like temporal logic, Petri nets, and compliance patterns [24].

With the rise of cloud computing, traditional centralized methods face scalability limits [18]. Emerging cloud-native BPC models—such as Security Validation as a Service (SVaaS) [25] and Compliance Monitoring as a Service (BP-MaaS) [20]—enable scalable, autonomous, and proactive compliance governance.

2.3. Microservices Architecture

Microservices Architecture (MSA) represents a major evolution from traditional monolithic systems, which bundle all functionalities within a single deployable unit. While monolithic designs were once effective for smaller systems, they become increasingly rigid and inefficient in distributed and rapidly changing environments [5]. Large monoliths suffer from limited scalability, increased complexity, and tight coupling, making updates or partial redeployments difficult and risky [26]. These systems often face technology lock-in, as all components must share the same technology stack, and they experience availability issues, since a failure in one part may disrupt the entire application [27].

To address these limitations, MSA decomposes applications into small, autonomous services, each focused on a single business capability and independently deployable [28]. It can be viewed as the second generation of Service-Oriented Architecture (SOA)—retaining the service-based concept while eliminating SOA's central bottlenecks like the Enterprise Service Bus (ESB) [29]. MSA instead employs lightweight communication through REST APIs or asynchronous message buses, enhancing modularity, cohesion, and system resilience [30].

Compared to SOA, microservices rely on event-based choreographies rather than orchestrations, enabling decentralized coordination and reducing dependencies between services [28]. This decoupling empowers autonomous development teams to update, deploy, and scale services independently, leading to improved uptime, continuous delivery, and faster innovation [5,27].

However, MSA introduces new challenges, such as complex service decomposition, testing and deployment overhead, and difficulty in coordinating cross-service features [27]. Despite these drawbacks, its cloud-native orientation makes it ideal for digital transformation and scalable cloud deployment on platforms like AWS, Azure, or Google Cloud [26].

For Business Process Compliance (BPC), MSA provides strong architectural alignment. Microservices are often deployed as containerized clusters in multicloud environments, supporting fine-grained monitoring and QoS- or regulation-aware compliance checks [31,32]. Moreover, MSA's adaptive and evolvable structure allows organizations to respond dynamically to regulatory or environmental changes through controlled automatic, semi-automatic, or global adaptation mechanisms [28].

In sum, while microservices introduce new coordination and testing challenges, they remain the preferred paradigm for achieving scalable, resilient, and compliant architectures in modern enterprise systems.

2.4. Cloud Technologies

The rapid growth of organizations and IT applications has dramatically increased both operational and infrastructure workloads, pushing many enterprises—particularly SMEs—to seek scalable, cost-effective alternatives to traditional on-premises systems [26]. Cloud computing emerged as a transformative solution, offering elastic scalability, reduced capital expenditure, and operational flexibility through a distributed network of virtualized computing resources [33].

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 December 2025 doi:10.20944/preprints202512.0916.v1

The launch of Amazon Web Services (AWS) in 2006 marked the beginning of the public cloud era, followed by rapid market expansion and global adoption [26,34]. The COVID-19 pandemic further accelerated this trend, making cloud infrastructure central to digital transformation and remote operations worldwide [34].

Cloud computing is fundamentally a distributed, virtualized environment where computational resources—servers, storage, and networking—are shared, reused, and provisioned dynamically on demand [33]. Its elastic and utility-like model allows organizations to scale up or down efficiently while paying only for consumed resources.

The emergence of cloud-native technologies has extended these principles further. As defined by the Cloud Native Computing Foundation [35] cloud-native systems are built to exploit the full potential of cloud environments through microservices, containers, service meshes, immutable infrastructure, and declarative APIs. These technologies promote resilience, scalability, and continuous delivery, enabling organizations to respond rapidly to evolving business needs.

Within this paradigm, Microservices Architecture (MSA) naturally complements cloud computing by distributing computing and storage across multiple services and nodes. This model supports deployment across public, private, and hybrid clouds, offering flexibility to migrate or replicate components seamlessly between environments [26].

In summary, cloud technologies—driven by virtualization, containerization, and microservices—have become foundational to modern enterprise transformation, enabling organizations to achieve agility, efficiency, and compliance in an increasingly dynamic digital landscape.

2.5. Related Work

With the foundational concepts defined, this section presents a systematic review of the related work. The literature on Business Process Compliance reveals a clear and logical progression that can be clustered into three main streams, moving from foundational concepts towards the challenges of modern, decentralized architectures.

2.5.1. Stream 1: Foundational Frameworks and Proactive Assurance

The initial wave of BPC research focused on establishing the need for systematic compliance management in distributed systems like Service-Oriented Architecture (SOA) and early Cloud environments. The primary concern was moving beyond manual, ad-hoc checks to more automated, proactive approaches [21,36]. This era placed a heavy emphasis on "shift-left" strategies, utilizing formal methods such as model-checking [25,37] and Petri nets [36] to verify compliance at design-time, before deployment.

A key development during this period was the introduction of holistic governance frameworks [21,38,39] that proposed central repositories for policies and a unified view of compliance. This centralized governance model laid the groundwork for the architectural pattern of "Compliance-as-a-Service", where compliance logic is decoupled into dedicated, reusable services [20,25,37].

2.5.2. Stream 2: Runtime Monitoring, Adaptation, and Lifecycle Management

As cloud environments grew more dynamic and complex, the research focus shifted from static, pre-deployment checks to the challenges of continuous, runtime compliance [19]. Researchers in this stream began adopting more powerful runtime technologies like Complex Event Processing (CEP) for real-time analysis [20,40] and the introduction of semantic, ontology-based models for richer, context-aware monitoring [14].

Researchers also focused heavily on the management of the entire compliance lifecycle. Researchers began to address the evolution of rules [41], the impact of process changes [42], and the need for unifying abstractions like "Compliance Descriptors" to manage this complexity [43,44]. This led to the concept of "Adaptive Compliance", aiming for systems that could dynamically respond to

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 December 2025 doi:10.20944/preprints202512.0916.v1

changes and violations, often using control loops like MAPE-K [19], particularly in architectures for cloud service brokers [32].

2.5.3. Stream 3: Decentralization, Operational Reality, and Next-Generation Challenges

The most recent research grapples with the challenges posed by highly dynamic, decentralized, and operational-heavy environments, particularly those defined by Microservices Architecture (MSA) and multi-cloud setups. MSA, which decomposes applications into small, independently deployable services [5], offers agility but exacerbates the difficulties of centralized monitoring and control.

This stream shows several key trends. First, researchers are the exploration of decentralized trust mechanisms, such as blockchain and distributed ledgers, to create immutable and transparent audit trails [13,31]. Second, many are move to address compliance at the operational and infrastructure level, giving rise to the powerful concept of "compliance by default" for infrastructure configuration [45] and the detection of operator errors [16]. Finally, there is a growing recognition that compliance is not just a technical problem but an economic one, leading to data-driven, post-mortem analysis using Process Mining [16] and the development of models for assessing the economic impact of compliance measures [46].

These streams show a clear evolution toward decentralized and operational-heavy environments. In Section 5.7, we will evaluate our proposed design against the specific approaches in this state-of-the-art to confirm its novelty and demonstrate how it addresses the field's persistent challenges.

2.6. Formal Foundations for BPC in MSA/Cloud

To rigorously define the compliance problem in a distributed, process-centric environment, we establish the following formal foundations. These definitions are aligned with established BPM and workflow literature. The "MSA/Cloud" context is explicitly reflected in the distributed nature of these definitions, where data (events) originates from multiple, independent services rather than a single monolithic source. These definitions describe the general artifacts and functions instantiated by our proposed framework in Section 4.

Let U_{case} be the universe of case identifiers, U_{act} be the universe of activity identifiers, U_{ts} be the universe of timestamps, U_{att} be the universe of attribute names, and U_{val} be the universe of data values.

Definition 2.6.1 (Event): *An event* e *is a tuple* $e = (c, a, t, \pi)$*, where:*

- $c \in U_{case}$ is the **case identifier**, linking the event to a specific process instance.
- $a \in U_{act}$ is the **activity identifier** (e.g., 'Approve Loan').
- $t \in U_{ts}$ is the **timestamp**.
- $\pi: U_{att} \rightharpoonup U_{val}$ is a partial function representing the data **payload**. In an MSA/Cloud context, this payload is critical for traceability and typically contains attributes like *serviceID* (identifying the microservice source) and *correlationID* (linking distributed events).

Let $\mathcal{E}$ be the universe of all possible events.

Definition 2.6.2 (Trace): *A distributed trace* τ *is a finite, non-empty sequence of events* $\tau = \langle e\_1, e\_2, \dots, e\_n \rangle$ *where* $e_i \in \mathcal{E}$*, and:*

- $\forall i, j \in [1..n]: e_i.c = e_j.c$ (all events belong to the same case). A trace is considered "distributed" as its constituent events e_i may originate from numerous independent services
- $\forall i \in [1..n-1]: e_i.t \leq e_{i+1}.t$ (events are ordered by timestamp).

In an MSA/Cloud context, a trace represents a reconstruction of a single process instance (a case). While formally defined by a shared case identifier ($e_i.c$), this set of events is practically assembled

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 December 2025 doi:10.20944/preprints202512.0916.v1

from numerous, independent services by a logging function (f_{log}) using correlation attributes (like *correlationID* from Def. 2.6.1) found in the event payload (π).

Let $\mathcal{T}$ be the universe of all traces.

Definition 2.6.3 (Event Log): *An event log* L *is a set of distributed traces,* $L \subseteq \mathcal{T}$.

Definition 2.6.4 (Process Model): *A process model* P *is a tuple* $P = (N, E, \lambda, N_s, N_e)$, *where:*

- N is a set of nodes (model elements).
- $E \subseteq (N \times N)$ is a set of control-flow edges.
- $\lambda: N \nrightarrow U_{act}$ is a partial labeling function mapping nodes to activity identifiers.
- $N_s \subseteq N$ is the set of start nodes.
- $N_e \subseteq N$ is the set of end nodes.

Definition 2.6.5 (Regulation and Policy Artifacts):

Regulation Source (R_{src}): An unstructured or semi-structured artifact (e.g., a legal document).

Compliance Rule (r): A formal, machine-readable predicate over a distributed trace, $r: \mathcal{T} \rightarrow$ {true, false}.

Compliance Policy ($\mathcal{C}$): A finite set of compliance rules, $\mathcal{C} = \{r_1, r_2, \ldots, r_k\}$.

Definition 2.6.6 (Policy Formalization Function): A policy formalization function f_{form} is a function that maps an unstructured regulation source to a formal compliance policy. $f_{form}: \mathcal{R}_{src} \rightarrow \mathcal{C}$

Definition 2.6.7 (Compliance Check Function): A compliance check function f_{check} (a form of conformance checking) is a function that takes a distributed trace and a policy and produces a verdict $v \in \{Compliant, NonCompliant\}$. $f_{check}: \mathcal{T} \times \mathcal{C} \rightarrow v$

Definition 2.6.8 (Logging Function): A logging function f_{log} is a function that subscribes to a stream of distributed business events ($\mathcal{E}^*$) and assembles them into a structured event log L. $f_{log}: \mathcal{E}^* \rightarrow L$

Definition 2.6.9 (Reporting Function): A reporting function f_{report} is an aggregation function that takes an event log and a policy to produce a compliance report R_{comp}. $f_{report}: L \times \mathcal{C} \rightarrow R_{comp}$

Definition 2.6.10 (Compliance Event): A compliance event $e_c \in \mathcal{E}$ is a specific type of event (generated by f_{check}) where $e_c.a =$ *'ComplianceCheckResult'* and its payload π contains $\{c, \mathcal{C}.id, v\}$.

Definition 2.6.11 (Identity Function): An identity function f_{id} is a function that maps an incoming request req (containing credentials or tokens) to an authenticated principal (a user or service) or to a null principal if authentication fails. $f_{id}: Request \rightarrow Principal \cup \{null\}$

These definitions (2.6.1-2.6.11) establish a precise, formal language for describing the data, artifacts, and functions within a distributed, process-centric compliance ecosystem. They provide the necessary foundation for the subsequent sections. With this formal model in place, we will now proceed to Section 3 to define the specific objectives for our solution, and then to Section 4 to introduce the BPC4MSA framework as a concrete architectural instantiation of these formal functions ($f_{form}, f_{check}, f_{log}, f_{\mathrm{id}}$, etc).

3. BPC4MSA Requirements Elicitation and Definition

This section details the methodology employed to derive the requirements for the BPC4MSA framework and presents the resulting requirements, which constitute the specific objectives for the solution being designed within our Design Science Research Methodology (DSRM) approach.

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 December 2025 doi:10.20944/preprints202512.0916.v1

3.1. Overall Research Methodology Context

As outlined in Section 1.2, this research adheres to a DSRM process [8,9]. The work presented in this section fulfills the *Define Objectives of a Solution* phase. Due to the significant challenges in achieving effective BPC within modern MSA/Cloud environments, particularly the lack of established reference architectures and patterns specific to this context [7], we undertook a rigorous requirements elicitation process. This process was informed by a preliminary analysis of the existing literature, which highlighted core problems, stakeholder needs, prevalent challenges, and proposed functionalities relevant to BPC in distributed and cloud settings.

3.2. Requirements Elicitation Context and Approach

The need for a novel BPC framework stems from several factors characterizing contemporary IT landscapes. These include escalating regulatory pressures across various domains, the strategic shift towards MSA to enhance organizational agility [43], the pervasive adoption of cloud computing platforms [36], the increasing prevalence of business processes operating across distributed systems and organizational boundaries [47], and the recognized need for automation to improve the efficiency, scalability, and reliability of compliance activities over manual approaches [36]. Domain-specific constraints, notably within banking [48] and healthcare (HIPAA), alongside overarching data privacy mandates such as General Data Protection Regulation (GDPR) further compound the complexity.

Our preliminary literature analysis underscored key research thrusts relevant to this context, including efforts in automating compliance certification [36], adapting compliance management for dynamic cloud environments [19,43], leveraging semantics for enhanced compliance reasoning [14], ensuring compliance throughout the system lifecycle [49,50], and addressing cloud adoption challenges for SMEs [51]. This body of work involves diverse stakeholders whose needs must be considered, prominently including Compliance Officers/Auditors [48], Business Users/Process Owners [14], and the Developers/Engineers responsible for building and maintaining the systems, alongside IT experts, end customers [36], regulators, and potentially cloud service providers [7].

Given this multifaceted context and the identified gap in comprehensive BPC solutions specifically architected for MSA/Cloud, a direct application of requirements elicitation techniques from more established domains (cf. [11]) was deemed insufficient. Therefore, we adopted a combined requirements derivation approach integrating:

First-Principles Analysis: To systematically derive functional and non-functional needs from the inherent properties and interactions of MSA, Cloud, and BPC, explicitly informed by the challenges documented in the literature.

Constructed Scenario Analysis: To explore and validate requirements within practical operational contexts, designing scenarios based on the specific stakeholder use cases identified in the literature.

This dual methodology ensures the resulting requirements possess both theoretical soundness and practical relevance necessary to guide the design of an effective BPC framework.

3.3. Derivation via First-Principles Analysis

The first-principles analysis component aimed to understand the fundamental compliance implications arising from the core technical and architectural characteristics of MSA and Cloud environments, independent of specific implementations but deeply informed by the systemic challenges reported in the literature. Our analysis proceeded in three main steps:

3.3.1. Identify Core Characteristics

We first enumerated and defined the salient characteristics of MSA (e.g., decentralization, independent deployability, service granularity, polyglot persistence, ephemerality, network communication reliance) [5,27], Cloud Computing (e.g., elasticity, on-demand resources, managed

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 December 2025 doi:10.20944/preprints202512.0916.v1

services, API-driven infrastructure, shared responsibility) [35] and the essential tenets of BPC (e.g., policy enforcement, auditability, non-repudiation, traceability, real-time monitoring) [4].

3.3.2. Analyze Interactions and Conflicts

Next, we analyzed the pairwise and multi-way interactions among these characteristics to identify inherent conflicts, tensions, or potential synergies impacting compliance. Crucially, this analysis was guided by the challenges highlighted in our literature review (Section 2.5), particularly the shift to runtime monitoring (Stream 2) and the complexities of decentralized, operational-heavy environments (Stream 3). This analysis focused on interactions relevant to:

- The difficulty of managing distributed data consistently across independent microservice databases versus BPC's need for strong data governance (e.g., privacy, lineage) [48]
- The complexity of monitoring and verifying compliance for complex, potentially asynchronous inter-service communication patterns [47].
- The friction between MSA's typical lack of centralized control and the need to enforce consistent compliance standards across independently developed services [52].
- The increased overall system complexity inherent in MSA impacting the ability to achieve holistic compliance understanding and verification [14].
- Deficiencies in observability and monitoring capabilities when trying to capture sufficient data for compliance verification across distributed services [47].
- Problems in defining clear boundaries and responsibilities for compliance in processes spanning multiple services or teams, especially in collaborative settings [53].
- The difficulty of maintaining compliance amidst the constantly evolving landscape of frequent microservice updates [19].
- The challenge of formalizing high-level compliance requirements into specific, verifiable rules at the microservice level [14].
- The potential for significant performance overhead when implementing runtime compliance checks [54].

By systematically exploring the interplay between MSA/Cloud characteristics (like decentralization, network reliance, ephemerality) and BPC needs (like auditability, traceability, policy enforcement) through the lens of these documented challenges, we identified the fundamental points of friction and opportunity.

3.3.3. Deduce Necessary Capabilities

Based on the specific conflicts and tensions identified in the previous step – which correspond to the literature challenges – we then deduced the fundamental system capabilities required for any BPC framework to effectively function within this context. These capabilities represent necessary architectural responses:

- To address the conflicts arising from decentralization and the lack of centralized control impacting auditability, the capability for reliable, distributed event correlation and aggregation was deduced as essential [38,55].
- To manage data governance across distributed data management complexities [48], capabilities for policy definition applicable to data handling and mechanisms to trace data flow across services were identified as necessary.
- To ensure traceability despite complex inter-service communication patterns and potential observability gaps [47], the capability for robust context propagation and comprehensive event logging across service boundaries was deemed fundamental.
- To counter the challenge of cloud ephemerality impacting audit trail continuity (related to the evolving landscape discussed by [19], capabilities for decoupled, persistent, and potentially immutable logging services were identified as critical.

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 December 2025 doi:10.20944/preprints202512.0916.v1

- To mitigate performance overhead concerns, the analysis suggested that compliance checking mechanisms should ideally be lightweight, distributable, and potentially operate asynchronously where feasible [56].
- To address the difficulty in formalizing requirements and managing complexity, capabilities supporting expressive yet processable policy languages, potentially leveraging semantic representations, were seen as necessary [14].
- To manage the evolving landscape and issues with defining boundaries, capabilities related to dynamic policy updates and clear association of rules with services/processes were deduced [19].

These deduced capabilities, arising directly from the interplay of core architectural principles and known compliance difficulties documented in the literature, formed the initial, theoretically grounded set of requirements for BPC4MSA.

In summary, the first-principles analysis allowed us to move beyond surface-level problems and identify a core set of essential capabilities mandated by the fundamental nature of MSA, Cloud, and BPC interactions, explicitly informed by and responding to the systemic challenges reported in the literature, thus providing a foundational layer for the requirements definition process.

3.4. Derivation via Constructed Scenario Analysis

To complement the theoretical deductions and ensure practical relevance, we designed and analyzed a set of realistic operational scenarios.

3.4.1. Scenario Design

We developed five distinct scenarios, each illustrating plausible yet challenging BPC situations grounded in common MSA/Cloud patterns and regulatory contexts (e.g., finance, healthcare data processing). These scenarios were designed to reflect specific stakeholder goals identified in the literature:

Scenario 1 - High-Volume Asynchronous Transaction Auditing (Compliance Officer Goal): This scenario detailed a process involving a customer initiating an online payment that triggered interactions across an Order Service, a Payment Gateway Service, a Fraud Detection Service, and an Account Service. The compliance requirement is not to block the transaction, but to ensure that a complete, verifiable audit trail is generated and checked for every transaction within a guaranteed time window (e.g., 5 minutes). The core challenge is ensuring this auditing system can handle peak transactional loads (e.g., Black Friday) without crashing, dropping events, or impacting the performance of the core transaction services. The audit must be guaranteed and eventual, even if processed seconds or minutes after the transaction completes. This reflects the need for resilient, high-throughput auditing rather than brittle, synchronous enforcement [22,47].

Scenario 2 - Auditing GDPR Right-to-Erasure Across Services (Auditor Goal): This scenario depicted an automated process triggered by a user's request to be forgotten under GDPR. It required coordinating deletion or anonymization tasks across a Customer Profile Service, an Order History Service, and a Marketing Preferences Service, potentially involving asynchronous communication. The core challenge was ensuring complete removal across all services and generating an immutable, verifiable audit trail proving compliant execution for the auditor [48,53].

Scenario 3 - Pre-deployment Compliance Check in CI/CD (Developer Goal): This scenario focused on a developer committing code for a new version of a Shipping Service. The CI/CD pipeline needed to automatically check the service's configuration and dependencies against organizational security policies (e.g., allowed base images, secret management) and data handling rules (e.g., ensuring no Personally Identifiable Information (PII) is logged) before deploying to a staging or production environment [48,55,57,58]

Scenario 4 - Dynamic Policy Update for Data Residency (Compliance Officer Goal): This scenario addressed the need to update a data residency policy (e.g., requiring data for European customers to remain within EU zones [59] due to a regulatory change. The challenge involved propagating this

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 December 2025 doi:10.20944/preprints202512.0916.v1

updated policy to multiple services (e.g., User Profile Service, Data Analytics Service) dynamically and verifying that subsequent processing adhered to the new constraint without service interruptions [19].

Scenario 5 - Business User Compliance Dashboard Interaction (Business User Goal): This scenario described a Business Process Owner for 'Order Fulfillment' reviewing a compliance dashboard. An alert indicated unusual delays violating an internal SLA policy within the process, traced to a specific interaction between the Inventory and Shipping services. The scenario detailed how the user navigated the dashboard to understand the violation context and initiated a pre-defined escalation workflow [48,60–62].

Each scenario explicitly defined the context, actors, triggers, sequence of operations (including potential parallel or asynchronous steps typical of MSA), data involved, specific compliance rules derived from regulations or internal policy, and explicitly identified potential points where compliance could fail.

3.4.2. Scenario Analysis

We systematically analyzed each constructed scenario using a combination of techniques:

Walkthroughs: We conducted detailed walkthroughs of each scenario, tracing the flow of control and data between microservices, identifying decision points, and mapping interactions against the specified compliance constraints.

Capability Gap Analysis: For each step, particularly potential failure points, we assessed the capabilities of typical MSA/Cloud infrastructure elements (e.g., API gateways, service meshes, standard logging libraries, orchestrators) and common operational practices. We identified where these standard capabilities would be insufficient to automatically detect or prevent the specific compliance violation described in the scenario. This highlighted gaps related to, for instance, lack of cross-service context propagation, inadequate granularity in logging, difficulty in enforcing complex temporal rules at runtime, or absence of integrated policy checking points.

Requirements Elicitation: The identified capability gaps directly informed the elicitation of concrete requirements for the BPC4MSA framework. For example, the gap in tracing data across asynchronous communication in Scenario 2 led to a requirement for robust event correlation mechanisms (FR3). The inability of standard CI/CD tools to perform deep policy checks in Scenario 3 led to requirements for design-time validation capabilities (FR2) and integration interfaces (FR10). The need for business-level visibility in Scenario 5 drove requirements for user-centric reporting and dashboards (FR7) and potentially low-code configuration (FR12). This process ensured that elicited requirements were directly tied to addressing specific, practical compliance challenges within the target environment.

3.5. Synthesized BPC4MSA Requirements

3.5.1. Overview

The analyses from the preceding sections, integrating first-principles reasoning with insights from constructed scenarios grounded in the literature, led to on a set of core functional capabilities and essential non-functional qualities required for the BPC4MSA framework. These synthesized requirements, refined from the deduced needs and practical challenges, became the design objectives for the artifact. They are formulated to address the unique complexities of achieving BPC within MSA/Cloud architectures, informed by the challenges and potential solutions identified in the literature. This section outlines these objectives, first describing the core functional capabilities the framework must possess, and second, the essential non-functional qualities it must exhibit. Within each description, the specific underlying requirements (FRs/NFRs) derived from the analysis are highlighted. Acknowledging the evolving technological landscape, forward-looking considerations regarding AI integration are also included.

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 December 2025 doi:10.20944/preprints202512.0916.v1

3.5.2. Core Functional Capabilities

The BPC4MSA framework must provide the following fundamental capabilities:

C1 - Comprehensive Policy Lifecycle Management: Addressing the challenge of consistently defining, updating, and applying compliance rules across numerous, independently deployed, and potentially heterogeneous microservices, this capability encompasses the end-to-end management of policies within the MSA/Cloud contex. It requires mechanisms for Formal Policy Specification (FR1) using machine-readable formats, potentially enhanced by Semantic Representation Support (FR11) using ontologies or similar structures to handle complexity and ambiguity [14,48]. It must also include robust Compliance Change Management (FR8) features to handle policy evolution, impact assessment, and dynamic propagation of updates across distributed services [19]. To facilitate reuse and standardization, support for a Compliance Pattern Library (FR13) is necessary [58,63].

C2 - Proactive Compliance Assurance: To prevent compliance violations from entering the dynamic microservices environment via frequent, independent service deployments, the framework must enable verification before execution. This capability is primarily realized through Design-Time Compliance Validation (FR2), allowing analysis of service designs, configurations, or code against policies, often integrated with development tools or CI/CD pipelines [48,57].

C3 - Runtime Compliance Monitoring and Auditing: Given that business processes in microservices execute across distributed services, often involving complex and asynchronous communication patterns, this capability addresses the need for continuous compliance oversight during execution. It necessitates Distributed Event Collection & Correlation (FR3) mechanisms to capture relevant activities across microservices and reconstruct process instances ([47]). Building on this, Runtime Compliance Monitoring & Auditing (FR4) is required to evaluate behavior against rules in a decoupled, asynchronous manner and trigger verification actions (e.g., logging for post-hoc review). This capability focuses on detecting and recording violations for post-execution analysis, rather than synchronous, real-time prevention.

C4 - Robust Compliance Auditing and Analysis: To overcome the inherent fragmentation of execution logs and data across decentralized microservices and reconstruct reliable evidence for verification and learning, the framework must provide comprehensive post-execution capabilities. This includes Automated Conformance Checking (FR5) to compare actual execution traces against expected models or rules [53], supported by Immutable Audit Trail Management (FR6) ensuring secure, reliable, and tamper-evident logging, potentially using technologies like blockchain [53,64]. To understand deviations, Root Cause Analysis Support (FR9) features are needed [48]. Findings must be communicated effectively through Compliance Reporting and Visualization (FR7) via dashboards [60,61], which should ideally offer Low-Code/No-Code Configuration (FR12) options for usability [62].

C5 - Seamless Ecosystem Integration: To function effectively within the complex and interconnected MSA/Cloud ecosystem (including diverse PAIS, observability tools, and CI/CD platforms), the framework cannot operate in isolation; therefore, it requires capabilities for Integration Interface (FR10) provision. This involves well-defined APIs and mechanisms to connect with external regulatory sources, PAIS, CI/CD platforms, observability tools, service meshes, and potentially identity management systems [31,65–67].

3.5.3. Essential Non-Functional Qualities

The practical success of BPC4MSA hinges on exhibiting key non-functional qualities:

NQ1 - Performance and Resource Efficiency: The framework must operate efficiently within the demanding MSA/Cloud context. This primarily requires high Scalability (NFR1) to handle dynamic loads and Performance Efficiency (NFR2) to ensure compliance checks impose minimal overhead on business processes. Managing latency, especially from potential AI integrations (NFR9), is also encompassed here [27].

NQ2 - Trustworthiness and Security: As a system underpinning compliance, trustworthiness is paramount. This fundamentally demands robust Security (NFR4) measures protecting the

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 December 2025 doi:10.20944/preprints202512.0916.v1

framework and its sensitive data [59] and ensuring High Availability & Resilience (NFR3) against failures [27]. Aspects like the reliability of potential AI outputs (NFR10), data privacy (NFR11), and the explainability of compliance decisions (NFR8) are critical sub-dimensions contributing to overall trustworthiness.

NQ3 - Adaptability and Usability: The framework must be practical to deploy, use, and evolve over time. Key aspects include Extensibility (NFR6) to accommodate new regulations and technologies [19] and Usability (NFR5) for diverse stakeholders, potentially enhanced by Low-code interfaces (FR12). Maintainability (NFR7) is also crucial for long-term viability in agile environments [27].

3.5.4. Summary of Core Objectives

In essence, the primary objective defined by these requirements is to create a BPC framework (BPC4MSA) that delivers the core functional capabilities (C1-C5) while exhibiting the essential non-functional qualities related to performance/efficiency (NQ1), trustworthiness/security (NQ2), and adaptability/usability (NQ3) necessary for effective operation within complex, dynamic, and distributed MSA/Cloud environments. The framework must address the unique challenges posed by these modern architectures while delivering actionable compliance insights and controls, with considerations for future AI integration. Section 4 will now detail the design of this framework, showing how it instantiates the formal functions (e.g., f_{check}, f_{log}) defined in Section 2.6 to meet these objectives.

4. The BPC4MSA Framework Design

This section presents the design of the BPC4MSA framework, the primary artifact developed in this research. The design directly addresses the core functional capabilities (C1-C5) and essential non-functional qualities (NQ1-NQ3) identified as objectives in Section 3. Furthermore, this architecture serves as a concrete instantiation of the formal functions ($f_{form}, f_{check}, f_{log}$, etc.) and data artifacts (L, $\mathcal{C}$, P) defined in Section 2.6.

4.1. Overview and Design Principles

BPC4MSA is conceived as a cloud-native, microservices-based framework designed for agility, scalability, and comprehensive compliance management. Key design principles include:

- Decentralization: Aligning with MSA principles, core compliance functionalities are decomposed into independent microservices.
- Event-Driven Communication: An event bus facilitates asynchronous communication and loose coupling between services, enhancing resilience and scalability.
- Separation of Concerns: Services are grouped logically based on their primary function (e.g., BPC Automation, System Management, Integration).
- API-Centricity: An API Gateway manages external communication, providing a unified interface for various client applications.
- Extensibility: The architecture is designed to accommodate future requirements, including potential integration with advanced AI capabilities.

The overall architecture, illustrated in **Figure 1**, depicts the relationships between client applications, infrastructure components, and the core microservice clusters.

4.2. Architecture and Components

The BPC4MSA framework comprises several interacting components, organized into logical clusters deployed server-side, interacting via an API Gateway and an Event Bus, and serving various client applications.

Formally, the BPC4MSA framework ($\mathcal{F}$) is a concrete instantiation of the concepts defined in Section 2.6. We define the framework as a tuple:

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 December 2025 doi:10.20944/preprints202512.0916.v1

$$\mathcal{F} = (\mathcal{S}_{auto}, \mathcal{S}_{mgmt}, \mathcal{S}_{int}, \mathcal{B}, \mathcal{G})$$

where:

- $\mathcal{S}_{auto}$ is the set of BPC Automation services, which instantiate the core compliance functions $(f_{form}, f_{check}, f_{report})$.
- $\mathcal{S}_{mgmt}$ is the set of System Management services, which instantiate the operational functions (f_{log}, f_{id}).
- $\mathcal{S}_{int}$ is the set of Integration services, which provide access to external artifacts $(R_{src},\ P)$.
- $\mathcal{B}$ is the Event Bus, the mechanism for handling the event stream $(\mathcal{E})$.
- $\mathcal{G}$ is the API Gateway, the unified interface to the framework's functions.
- The components of this framework are detailed below and illustrated in **Figure 1**.

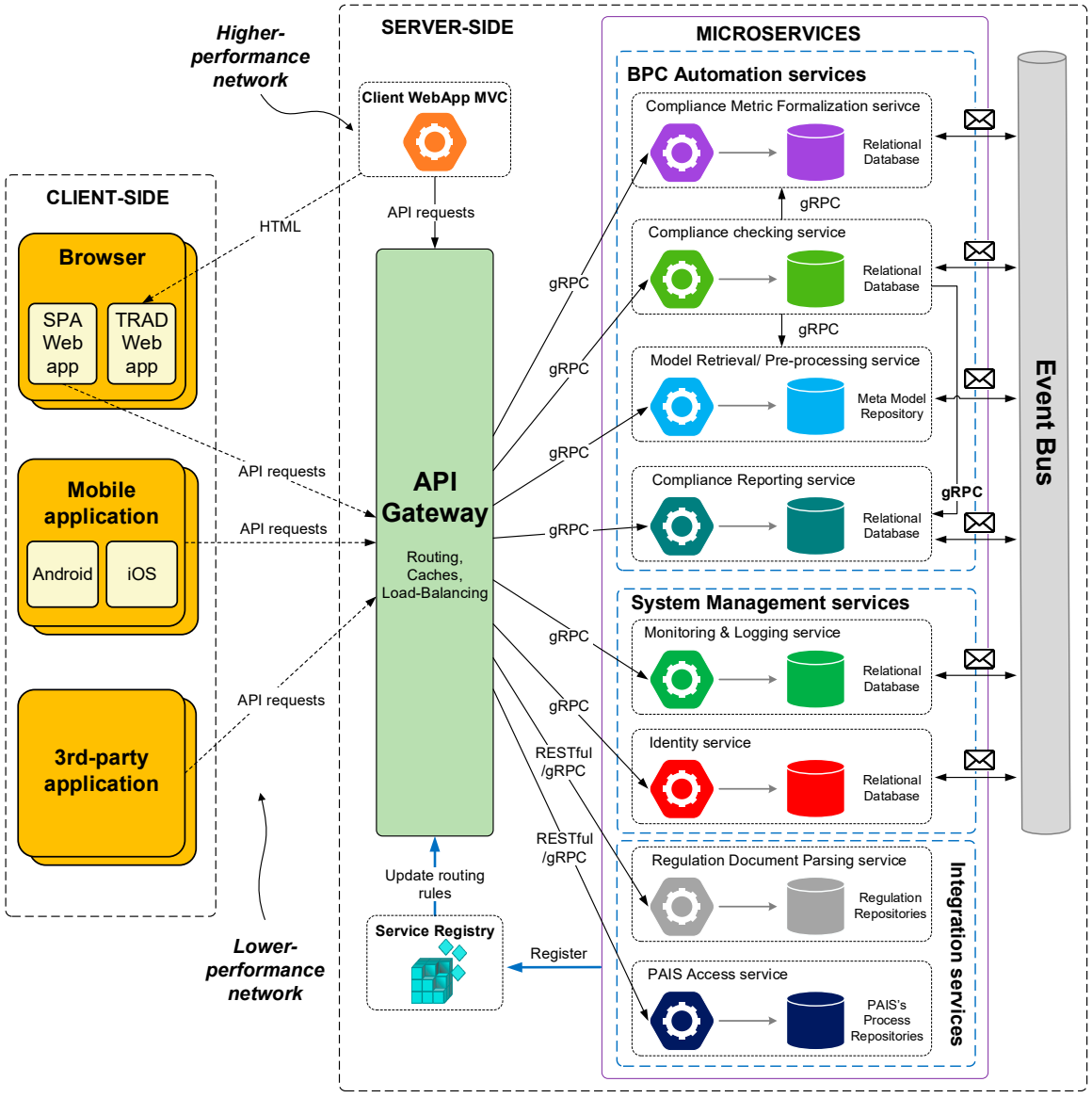

Figure 1. Realization of the BPC4MSA.

4.2.1. Client-Side Applications

These represent the consumers of the framework's functionalities and interact with the server-side via the API Gateway using standard protocols like RESTful APIs. Examples include:

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 December 2025 doi:10.20944/preprints202512.0916.v1

- Web Applications (SPA/TRAD): Providing interfaces for compliance officers, auditors, business users, and potentially developers to define policies, monitor status, view reports, and manage violations.
- Mobile Applications: Offering potentially optimized views and interactions for on-the-go monitoring or approvals.
- Third-Party Applications: Allowing external systems (e.g., Governance, Risk, and Compliance (GRC) platforms, external auditing tools) to integrate with BPC4MSA via public APIs.

4.2.2. Infrastructure Components:

These provide essential cross-cutting concerns for managing and operating the microservices:

- API Gateway: Acts as the single entry point for all client requests. It handles routing, load balancing, caching, and potentially initial authentication/authorization, abstracting the underlying microservice structure from clients.
- Service Registry: Enables dynamic service discovery, allowing services and the API gateway to locate instances of other services in the potentially dynamic cloud environment. Services register themselves upon startup.
- Event Bus: Facilitates asynchronous, publish-subscribe communication between microservices. This decouples services, allowing them to react to events (e.g., policy updates, detected violations, process milestones) without direct dependencies, enhancing resilience and scalability.

4.2.3. Microservice Clusters

The core logic resides in distinct microservices grouped by function:
BPC Automation Services ($\mathcal{S}_{auto}$): These services implement the primary compliance logic:

- Compliance Metric Formalization Service: (Instantiates f_{form}) Responsible for translating compliance requirements (potentially received from the Regulation Document Parsing service or user input via client apps) into the formal representation used by the checking engine (e.g., BhR, PCL). This service could potentially interface with AI models in the future to assist in translating natural language policies (related to C1).
- Model Retrieval/Pre-processing Service: (Retrieves P) Fetches business process models and related metadata from external sources (via the PAIS Access service) and prepares them for compliance analysis (related to C1, C5).
- Compliance Checking Service: (Instantiates f_{check}) The core engine that performs compliance verification. It takes formalized rules/metrics and process models/traces as input and evaluates compliance, potentially supporting both design-time checks (C2) and runtime/post-execution analysis (C3, C4) based on events from the Event Bus or data from the Monitoring & Logging service.
- Compliance Reporting Service: (Instantiates f_{report}) Aggregates compliance results, generates reports, and provides data for dashboards presented via client applications. It could potentially leverage AI in the future for generating natural language summaries or explanations of compliance status or violations (related to C4).

System Management Services ($\mathcal{S}_{mgmt}$): These provide operational support:

- Monitoring & Logging Service: (Instantiates f_{log}) Collects operational logs and metrics from all microservices for health monitoring and performance analysis. Crucially, it also aggregates and stores the *compliance audit trail* data (events, check results) generated by other services, potentially via the Event Bus, ensuring data is available for auditing (related to C3, C4).
- Identity Service: (Instantiates f_{id}) Manages authentication and authorization for users and potentially services interacting with the framework, ensuring secure access control (related to NQ2).

Integration Services ($\mathcal{S}_{int}$): These handle communication with external systems:

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 December 2025 doi:10.20944/preprints202512.0916.v1

- Regulation Document Parsing Service: (Retrieves R_{src}) Interfaces with external repositories or tools to retrieve and potentially parse regulatory documents or external policy sources, feeding information to the Formalization service (related to C1, C5).
- PAIS Access Service: (Retrieves L or $\mathscr{E}^*$ from external systems) Provides a dedicated interface for interacting with various Process-Aware Information Systems (PAIS) to retrieve process models and execution data (related to C5).

Each microservice typically manages its own dedicated database (e.g., Relational, NoSQL, Meta Model Repository, Regulation Repositories) to maintain loose coupling and allow for technology choices optimized for the service's specific needs. Communication *between* microservices often utilizes efficient protocols like gRPC for synchronous requests where needed, while leveraging the Event Bus for asynchronous notifications and data propagation.

4.3. Addressing Requirements

We now demonstrate how the BPC4MSA architecture and its components fulfill the core functional capabilities (C1-C5) and address the essential non-functional qualities (NQ1-NQ3) defined as objectives in Section 3.5. This analysis provides the explicit link between the requirements (Section 3), the formal foundations (Section 2.6), and the artifact design (Section 4).

4.3.1. Mapping Functional Capabilities

C1 (Policy Lifecycle Management): This capability is primarily realized through the interplay of the Compliance Metric Formalization Service and the Regulation Document Parsing Service. These services jointly instantiate the Policy Formalization Function ($f_{form}: \mathcal{R}_{src} \rightarrow \mathcal{C}$), which transforms an unstructured Regulation Source (R_{src}) into a formal Compliance Policy ($\mathcal{C}$) (FR1, FR11). This service would also manage a Compliance Pattern Library (FR13) to support reusable rule definitions. Policy evolution (FR8) is managed through updates to these services, with changes propagated via the Event Bus to the Compliance Checking Service.

C2 (Proactive Compliance Assurance): This is enabled by configuring the Compliance Checking Service to operate in a design-time mode. In this mode, it instantiates the Compliance Check Function (f_{check}), but applies it to a Process Model (P) (retrieved by the Model Retrieval Service) rather than a runtime trace (FR2).

C3 (Runtime Monitoring and Auditing): This is the core runtime capability. The Monitoring & Logging Service instantiates the Logging Function $f_{log}: \mathcal{E}^* \rightarrow L$, subscribing to the Event Bus to consume the stream of distributed Events ($\mathcal{E}^*$) and assemble them into a persistent Event Log (L). The Compliance Checking Service then instantiates the Compliance Check Function ($f_{check}: \mathcal{T} \times \mathcal{C} \rightarrow v$), applying a Policy ($\mathcal{C}$) to a Distributed Trace (τ) from the log (FR3, FR4). Violations trigger a Compliance Event (e_c) to be published back to the Event Bus for logging and reporting services.

C4 (Auditing & Analysis): This capability is built upon the outputs of C3. The Event Log (L) managed by the Monitoring & Logging Service serves as the immutable audit trail (FR6). The Compliance Reporting Service instantiates the Reporting Function ($f_{report}: L \times \mathcal{C} \rightarrow R_{comp}$), which aggregates data from the log to generate reports and dashboards (FR5, FR7, FR9, FR12). This architecture, by connecting the Compliance Reporting Service (C4) and the Compliance Metric Formalization Service (C1) via the Event Bus, theoretically proposes the mechanism to close the "Broken Feedback Loop" (GAP-2) mentioned in Section 1.1 and Section 5.7. It enables runtime insights from f_{report} to programmatically inform and update the policies managed by f_{form}. The empirical validation of this specific feedback loop mechanism is outside the scope of our preliminary prototype and is a key area for future work.

C5 (Ecosystem Integration): This is directly addressed by the dedicated Integration Services (PAIS Access, Regulation Document Parsing) and the API Gateway, which provide the necessary interfaces (FR10) to connect with external systems and data sources

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 December 2025 doi:10.20944/preprints202512.0916.v1

4.3.2. Addressing Non-Functional Qualities

NQ1 (Performance and Resource Efficiency): The microservices architecture inherently supports Scalability (NFR1) by allowing independent scaling of high-load services (e.g., Compliance Checking Service, Monitoring & Logging Service). Performance Efficiency (NFR2) is addressed through asynchronous, non-blocking communication via the Event Bus, a design choice whose trade-offs are empirically evaluated in Section 6. The management of potential AI-related latency (NFR9) would be localized to the specific services that integrate them, such as the Compliance Metric Formalization Service.

NQ2 (Trustworthiness and Security): Security (NFR4) is addressed through the dedicated Identity Service, secure communication protocols (e.g., TLS), network policies in the deployment environment, and secure data storage practices. High Availability & Resilience (NFR3) are achieved through the redundancy inherent in MSA deployments on platforms like Kubernetes and the decoupling provided by the Event Bus . Trustworthiness is further enhanced by immutable logging approaches (FR6) and the requirement for explainable outputs (NFR8), while AI integrations would necessitate specific validation (NFR10) and privacy measures (NFR11).

NQ3 (Adaptability and Usability): The modular microservice design promotes Extensibility (NFR6) and Maintainability (NFR7). For example, a new compliance rule can be added by updating the Compliance Metric Formalization Service without redeploying the entire system. Usability (NFR5) is addressed through the API Gateway, which provides a stable interface for user-facing client applications, including those offering low-code configuration (FR12)

4.4. Technology Stack

The microservices architecture allows for technology heterogeneity. However, a potential, coherent technology stack aligned with the requirements could include:

- Orchestration: Kubernetes (for deployment, scaling (NFR1), resilience (NFR3)).
- Event Bus: Apache Kafka or RabbitMQ (for scalable (NFR1), resilient (NFR3) asynchronous communication).
- API Gateway: Kong, Traefik, or cloud provider gateways (for managing external access, routing).
- Service Registry: Consul, etcd, or Kubernetes-native discovery (for dynamic discovery).
- Databases: PostgreSQL (for relational data, e.g., Identity, Reporting), Cassandra/MongoDB (for high-volume logs/events, supporting NQ1), Graph Database (potentially for policy/model relationships supporting C1/FR11).
- Monitoring/Logging: Elasticsearch/Fluentd/Kibana (EFK) stack or Prometheus/Grafana (for observability, supporting C4, NQ3).
- Service Implementation: Go, Java (Spring Boot), Python (FastAPI) - chosen based on team expertise and specific service needs (contributing to NQ3/NFR7).
- Communication: gRPC (for efficient internal synchronous calls), RESTful APIs (for external interaction).

The choice of specific technologies should be driven by detailed performance (NQ1/NFR2), security (NQ2/NFR4), maintainability (NQ3/NFR7), and extensibility (NQ3/NFR6) requirements.

4.5. Deployment Considerations

BPC4MSA is designed for cloud deployment, leveraging cloud-native principles:

- Deployment Models: Can be deployed on managed Kubernetes services (e.g., EKS, GKE, AKS), self-managed Kubernetes clusters, or potentially adapted for serverless platforms where applicable. Hybrid deployments (part cloud, part on-premise) are feasible but increase complexity.

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 December 2025 doi:10.20944/preprints202512.0916.v1

- Scalability & Resilience: Achieved via Kubernetes auto-scaling based on resource utilization (CPU, memory) or custom metrics (e.g., queue length for the Event Bus), fulfilling NQ1/NFR1. Redundancy across multiple availability zones addresses NQ2/NFR3.
- Security: Relies on cloud provider security features (network policies, IAM), secure configurations (e.g., secrets management), authentication/authorization via the Identity Service, and secure communication protocols (TLS), addressing NQ2/NFR4. Data encryption at rest and in transit is essential.
- Data Management: Strategies for managing distributed data persistence, backups, and potential cross-region replication need careful consideration based on specific compliance requirements (e.g., GDPR data residency).

The deployment strategy must ensure that the non-functional qualities (NQ1-NQ3) identified in Section 3.5 are met within the chosen cloud environment.

5. Design Evaluation: Systematic Literature Review

This section presents the evaluation of the BPC4MSA framework design artifact (detailed in Section 4). Following established Design Science Research Methodology (DSRM) guidelines for artifact evaluation [8,9] and drawing methodological inspiration from [11], we employ a Systematic Literature Review (SLR). This method is specifically chosen as our design evaluation to validate the framework's conceptual novelty against the state-of-the-art, and is distinct from the prototype's preliminary evaluation presented in Section 6. The primary goal of this evaluation SLR is to rigorously analyze the relevant state-of-the-art in Business Process Compliance (BPC) for Microservices Architecture (MSA) and Cloud environments, positioning the BPC4MSA design against existing approaches, identifying its potential contributions, and assessing the extent to which its design addresses the requirements (Capabilities C1-C5, Qualities NQ1-NQ3) defined in Section 3.5 relative to prior work.

5.1. Methodology

The SLR was conducted following established guidelines to ensure rigor and transparency. The process involved defining research questions, executing a systematic search, applying a multi-stage screening protocol based on predefined criteria, extracting relevant data, and synthesizing the findings. The overall flow of study selection is depicted in the PRISMA diagram (see Figure 2)

5.2. Research Questions (RQs) for SLR

The SLR aimed to answer the following specific evaluation-focused research questions:

- RQ-SE1: What existing frameworks, architectures, methods, or tools specifically address BPC (including aspects like policy management, monitoring, auditing, enforcement) within MSA, Cloud-native, or closely related Cloud/SOA contexts?
- RQ-SE2: How do these existing approaches address the core functional capabilities (C1-C5) and non-functional qualities (NQ1-NQ3) identified as requirements for BPC4MSA in Section 3.5?
- RQ-SE3: What are the primary limitations or gaps in the current state-of-the-art concerning BPC for MSA/Cloud environments, based on the reviewed literature?
- RQ-SE4: How does the proposed design of BPC4MSA compare to existing approaches, and what are its potential novel contributions towards addressing the identified gaps (RQ-SE3) and requirements (RQ-SE2)?

5.3. Search Strategy

A systematic search was performed across major academic databases relevant to computer science and information systems, including Scopus and Web of Science (WOS). Searches were primarily conducted within titles, abstracts, and keywords, limited to publications in English. The search strategy involved constructing search strings based on keywords related to the core concepts:

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 December 2025 doi:10.20944/preprints202512.0916.v1

BPC, Frameworks/Architectures, and MSA/Cloud/SOA environments. To ensure comprehensive coverage, several variations were employed, combining core terms with optional refinements. Key search term variations included:

Search Term 1 - Platforms Focus: ("business process compliance" OR "process compliance" OR "workflow compliance") AND ("framework" OR "architecture" OR "system" OR "model" OR "approach" OR "method" OR "tool" OR "platform" OR "solution") AND ("microservice architecture" OR "microservices" OR "cloud-native" OR "cloud computing" OR "cloud environments" OR "cloud services" OR "SOA" OR "service-oriented architecture")

Search Term 2 - Requirements Focus: ("business process compliance" OR "process compliance" OR "workflow compliance") AND ("framework" OR "architecture" OR "system" OR "model" OR "approach" OR "method" OR "tool" OR "platform" OR "solution") AND ("requirements" OR "functional requirements" OR "non-functional requirements" OR "compliance requirements")

5.4. Study Selection and Screening Process

A multi-stage screening process, guided by the PRISMA methodology (see Figure 2), was employed to systematically identify relevant studies. After removing 240 duplicates from the initial 1,484 records, the titles and abstracts of 1,244 unique articles were screened. This screening excluded 1,197 records (436 by title and 761 by abstract), resulting in 47 articles for full-text analysis. After this analysis, 9 full-text articles were excluded for not meeting the eligibility criteria. This resulted in the final inclusion of 38 articles deemed most relevant for the evaluation synthesis.

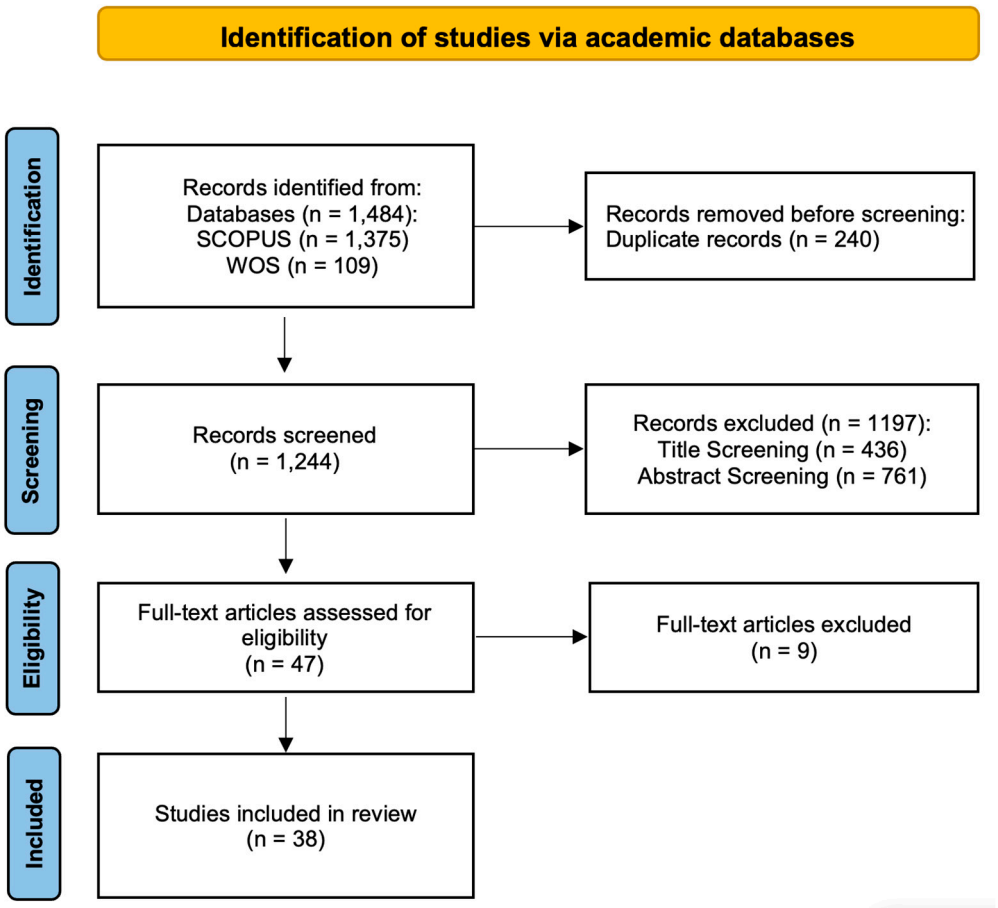

Figure 2. PRISMA flow diagram.

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 December 2025 doi:10.20944/preprints202512.0916.v1

5.5. Data Extraction

A predefined data extraction form was used to systematically collect information from each of the 38 included studies. To ensure consistency and rigor, the form was designed to capture key data points necessary to answer the evaluation research questions. These data points included: publication details (authors, year, title, venue), the stated problem or motivation, details of the proposed solution (e.g., framework name, architectural approach, key components), specific BPC aspects addressed (e.g., policy, monitoring, audit, enforcement), the explicit or implicit handling of requirements corresponding to BPC4MSA's capabilities (C1-C5) and qualities (NQ1-NQ3), the underlying technologies or formalisms used, the evaluation methods employed within the study, and any limitations or future work identified by the authors.

5.6. Synthesis Method

The extracted data was synthesized using a combination of thematic and comparative analysis. Thematic analysis was employed to identify, analyze, and report patterns (themes) within the data, which enabled a qualitative understanding of the state-of-the-art and the identification of research trends and persistent challenges (addressing RQ-SE1 and RQ-SE3). Subsequently, a comparative analysis was conducted to systematically map the features, contributions, and limitations of the approaches described in the selected literature against the design principles, architecture, and intended capabilities of the BPC4MSA framework. This direct comparison was instrumental in positioning BPC4MSA, assessing its novelty, and formulating the comparative matrix (see **Table 1**), thereby answering RQ-SE2 and RQ-SE4.

5.7. Literature Analysis & Framework Positioning

With the state-of-the-art established, we now evaluate the BPC4MSA framework's design (from Section 4) against the 38 selected papers. This comparative analysis begins by synthesizing the findings from the literature to answer RQ-SE1 (What existing approaches exist?) and RQ-SE2 (How do they address the defined requirements?). This synthesis reveals significant patterns in the focus of the BPC research community and provides the baseline against which we evaluate our artifact's novelty.

The following quantitative analysis (addressing RQ-SE1 and RQ-SE2) is based on the 38 papers of the comparative matrix (see **Table 1**):

- Core Capabilities are Widely Addressed: A vast majority of the analyzed literature addresses the core tenets of compliance management. Policy Lifecycle Management (C1) is considered in 89% of papers (34 out of 38), and Adaptability (NQ3) is a key concern in 89% (34 out of 38), indicating these are foundational research topics.
- Proactive and Reactive Measures are Prevalent: Both "shift-left" and "shift-right" approaches are very common. Proactive Compliance Assurance (C2) is addressed by 84% of papers (32 out of 38), and an equal percentage, 84% (32 out of 38), address Compliance Auditing & Analysis (C4).
- Runtime Monitoring is a key, but not universal, Focus: While still a majority concern, Runtime Monitoring & Enforcement (C3) is the least frequently addressed of the core capabilities, appearing in 63% of papers (24 out of 38). A qualitative look confirms the finding from GAP-3: most of these works focus on monitoring and detection, with true automated enforcement remaining rare.

Qualitatively, several patterns emerged:

- The Proactive vs. Reactive Divide: There is a clear pattern of mutual exclusivity between papers strong in Proactive Assurance (C2) and those strong in Compliance Auditing & Analysis (C4). Papers proposing formal, design-time verification methods [16,36,37] rarely include sophisticated post-mortem analysis, and vice-versa for papers focused on BI or process mining [16,68]. This further supports the "Broken Feedback Loop" (GAP-2).

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 December 2025 doi:10.20944/preprints202512.0916.v1

- The Rise of Holistic Frameworks: A noticeable trend in later years (post-2015) is the emergence of more comprehensive frameworks that attempt to address three or more of the core capabilities simultaneously [32,39,43,44,55,69,70]. This indicates a growing maturity in the field and a recognition that a holistic approach is necessary.
- Architectural Drivers: The architectural approach is a strong predictor of focus. Papers based on formal methods (Petri nets, model checking) are almost exclusively focused on C2. Papers based on CEP and event-driven architectures are focused on C3. Papers based on data warehousing and BI are focused on C4. This specialization contributes to the siloing of research and the identified gaps.

This analysis of the state-of-the-art (answering RQ-SE1 and RQ-SE2) allows us to evaluate the BPC4MSA framework's novelty (addressing RQ-SE4) and identify the specific gaps it addresses (RQ-SE3). We find that while many solutions address individual requirements, the literature lacks holistic frameworks that target the specific interplay of challenges in MSA/Cloud environments. Our evaluation confirms that the BPC4MSA design is novelly architected to address what we identify as five persistent research gaps, which the reviewed literature fails to resolve in an integrated manner:

- GAP-1: The "Last Mile" Problem: The process of translating ambiguous legal text into precise, machine-readable rules remains a largely manual, error-prone bottleneck, limiting the agility of the entire compliance lifecycle.
- GAP-2: The Broken Feedback Loop: There is a clear disconnect between proactive design-time tools, reactive runtime monitoring systems, and post-mortem analysis platforms, with no automated mechanism to feed insights back into design-time models.
- GAP-3: Weak or Absent Automated Enforcement: The vast majority of research focuses on monitoring and detection. True automated enforcement—actively preventing or correcting violations at runtime without human intervention—is rarely addressed.
- GAP-4: Integrating Internal (Architectural) and External (Regulatory) Compliance: The literature largely treats architectural integrity and regulatory compliance as separate problems, lacking a holistic framework that manages both simultaneously.
- GAP-5: Scalability and Practicality of Advanced Solutions: Many proposed solutions based on formal methods, semantic ontologies, or blockchain face significant practical hurdles, including high modeling overhead and performance bottlenecks.

The BPC4MSA framework is explicitly designed to address these consolidated gaps (addressing RQ-SE4). Its core design centers on a unified Policy Lifecycle Management (C1) to address GAP-1 and GAP-2. The Runtime Monitoring & Auditing (C3) capability is designed for active asynchronous verification (GAP-3). The Proactive Compliance Assurance (C2) validates both internal and external rules (GAP-4). Finally, by using a scalable architecture, its design confronts the challenges in GAP-5. **Table 1** provides a detailed comparative analysis of the reviewed literature against the BPC4MSA framework's capabilities.

Table 1. Comparative Matrix of BPC Literature.

ID	Citation	Architectural Approach	C1	C2	C3	C4	C5	NQ1	NQ2	NQ3
1	[71]	Remote auditing, trusted computing, secure logging	-	+/-	+/-	+	-	-	+	-
2	[36]	Petri nets, formal patterns	+/-	+	-	+/-	-	+/-	+	-
3	[31]	Blockchain, smart contracts	+/-	+/-	+	+	+/-	+/-	+	+/-
4	[20]	Service-based, Complex Event Processing (CEP)	+	+/-	+	+/-	+	+	+/-	+
5	[22]	Self-monitoring, process rewriting	+/-	+/-	+	+/-	+/-	+/-	+	+
6	[38]	SOA, CEP, DSLs	+	+	+	+	+	+	+	+
7	[37]	Cloud-based service, model-checking	+/-	+	-	+	+	+	+	+
8	[32]	Cloud service broker, legal/QoS framework	+/-	+/-	+	+/-	+/-	+/-	+/-	+

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 December 2025 doi:10.20944/preprints202512.0916.v1

9	[69]	Cloud service broker, autonomic computing (MAPE-K)	+	+	+	+/-	+/-	+/-	+/-	+
10	[25]	Cloud-based service, model-checking	+/-	+	-	-	+	+	+/-	+
11	[14]	Semantically-enabled, ontologies, COaaS	+	+	+	+	+/-	+/-	+/-	+
12	[44]	Model-Driven Architecture (MDA), Compliance Descriptors	+	+	+	+/-	+	+/-	+	+
13	[45]	Configuration management (Chef), experience report	+/-	-	+	+/-	+/-	+	+	+/-
14	[19]	Adaptive process (MAPE-K loop)	+	+/-	+	+/-	+/-	+/-	+	+
15	[72]	Goal-oriented Requirement Language (GRL), MCDA	+/-	+/-	-	-	-	-	+/-	-
16	[41]	ISC evolution, RETE algorithm	+	+/-	+	-	-	+/-	+/-	+
17	[23]	Variability Descriptors, Compliance Descriptors	+	+	+	+/-	+	+/-	+	+
18	[43]	Model-Driven Architecture (MDA), Compliance Descriptors	+	+	+	+/-	+	+/-	+	+
19	[16]	Process Mining, XES, economic assessment	-	-	-	+	+/-	+/-	+/-	+/-
20	[73]	Timed Automata, TCTL, UPPAAL model checker	+/-	+	-	-	-	-	+	-
21	[51]	Model-based, SPIN model checker, LTL	+/-	+	+	+/-	-	-	+/-	+/-
22	[74]	DSL, Model-Driven Development (MDD), CEP	+/-	-	+	-	+/-	+/-	+/-	+
23	[18]	User-centered, SPIN model checker, extended patterns	+	+	+	+/-	-	+/-	+	+
24	[21]	SOA, knowledge-based, BPCL	+	+/-	+	+	+	+/-	+	+/-
25	[55]	SOA, Deming cycle, KCIs, data warehouse	+/-	+	+/-	+	+/-	+/-	+/-	+
26	[75]	Integrated compliance graph, decision support	+/-	+	-	+/-	+/-	+/-	+	+
27	[42]	Integrated graph, interaction analysis	+/-	+	-	+/-	+/-	+/-	+	+
28	[76]	Integrated graph, process adaptation tool (BCIT)	+/-	+	-	+/-	+/-	+/-	+	+
29	[70]	State-aware, ROP ontology, Drools	+/-	-	+	+/-	+/-	+/-	+	+
30	[77]	MTL, rule elicitation, graphical representation	+	+	-	-	-	+/-	+	+/-
31	[78]	Semantic (OWL/SWRL), ontologies, Drools	+	+/-	-	+	+	+/-	+	+
32	[39]	End-to-end framework, VbMF, DSLs, CEP	+/-	+	+	+	+	+/-	-	+
33	[66]	DevSecOps, security checklist methodology	+	+	-	+/-	+	+/-	+	+
34	[40]	TOSCA, XML nets, CEP, anti-patterns	+/-	+	+	+/-	+/-	+/-	+	+/-
35	[79]	TOSCA, XML nets, CEP, anti-patterns	+/-	+	+	+	+	+	+	+/-
36	[46]	Economic assessment, Workflow Patterns, Cost-Benefit Analysis	-	+/-	-	+/-	-	+/-	+/-	+/-
37	[68]	Business Intelligence, Data Warehouse, OLAP	-	-	-	+	+/-	+/-	+	+/-
38	[50]	Adaptive SOA, AOP, CEP	+/-	-	+	+/-	+/-	-	+/-	+

**Legend*: *+ Fully and explicitly addressed; +/- Partially addressed or addressed implicitly; - Not addressed or not the focus of the work.*

5.8. SLR Summary and Novelty Confirmation

The systematic literature review substantiates the initial problem statement regarding the challenges of BPC in contemporary MSA/Cloud environments. The comparative analysis reveals that

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 December 2025 doi:10.20944/preprints202512.0916.v1

while existing approaches address certain aspects of compliance, significant gaps remain (GAP-1 to GAP-5), particularly a lack of holistic frameworks offering integrated, end-to-end support.

Against this backdrop, the proposed BPC4MSA framework demonstrates clear novelty. Its design directly targets the identified research gaps by proposing an integrated, MSA/Cloud-native architecture that handles distribution, dynamism, and scale. This evaluation, therefore, validates the relevance and conceptual soundness of the BPC4MSA design. It provides strong support for the design's novelty and establishes the foundation for the second part of our evaluation: the empirical study of the artifact's practical performance. This systematic review confirms the conceptual novelty and relevance of the BPC4MSA design. To assess the practical feasibility of this architecture, the following section details a prototype implementation and its preliminary evaluation.

6. BPC4MSA Prototype: Implementation and Preliminary Evaluation

To evaluate the practical utility and performance of the BPC4MSA artifact against the non-functional requirements for Performance (NQ1) and Adaptability (NQ3), we implemented a prototype and conducted a rigorous comparative empirical study. This study was designed to answer the following research questions:

- RQ-AE1: How do the architectures compare in terms of performance and resource efficiency under increasing load? (Evaluates NQ1)
- RQ-AE2: How do the architectures differ in their stability and resilience when pushed to their performance limits? (Evaluates NQ1, NQ3)

6.1. Experimental Setup

To answer the artifact evaluation research questions (RQ-AE1 and RQ-AE2), three functionally identical but architecturally distinct systems were implemented for comparison. All three architectures implement an identical loan application business process. A consistent technology stack (Python 3.9, FastAPI, PostgreSQL) was used, and all systems were containerized using Docker and orchestrated via Docker Compose to ensure environmental consistency. The architectural pattern itself was isolated as the primary independent variable.

To ensure a fair basis for performance comparison, a simulated function was integrated into the compliance-checking step of all three architectures. This function simulates a computationally intensive, CPU-bound task by implementing a recursive 0-1 knapsack problem with n=30 items. This algorithm, which has an exponential time complexity of $O(2^n)$, was chosen specifically to represent the NP-hard nature of many formal compliance-checking problems [80]. This design ensures the bottleneck under test is computational processing and architectural overhead, not external I/O latency, thus ensuring a consistent and non-trivial processing load for each compliance check. This simulated workload represents the computational cost of executing the Compliance Check Function (f_{check}), as formally defined in Definition 2.6.7, against a complex policy. The complete source code and experimental scripts are publicly available for reproducibility[1].

The three tested architectures are defined as follows:

- BPC4MSA (Event-Driven): The full prototype implementation of our proposed framework, featuring asynchronous, event-driven communication via a message bus (Kafka) and decoupled microservices for business logic, compliance checking, and auditing.
- Synchronous SOA: A baseline architecture simulating a traditional Service-Oriented Architecture. In this version, the compliance check is a synchronous, blocking HTTP call that must complete before the business process can proceed.
- Monolithic: A baseline representing a traditional all-in-one application. All business logic, compliance checking, and data persistence are handled within a single, tightly-coupled service.

[1] GitHub Repository: https://github.com/hangoclong/bpc4msa.git

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 December 2025 doi:10.20944/preprints202512.0916.v1

All three architectures were implemented using Python and FastAPI and were containerized using Docker. They implement identical business logic (a loan application process) and the same set of compliance rules, ensuring that the only variable under test is the architectural pattern.

6.2. Experiment Procedures and Data Collection

The experimental procedure was automated and orchestrated using bash scripts (e.g., run_experiment.sh) to ensure a repeatable process for system deployment, test execution, and data collection.

The primary experiment subjected each architecture to a stepped-load test using the Locust open-source load testing framework. The load profile (defined in locustfile_experiment1_baseline.py) simulated a ramp-up from 5,000 to 20,000 concurrent users submitting loan applications. During each test run, two categories of data were collected:

- Performance Metrics: Throughput (requests per second) and response latency (average and 95th percentile) were captured by the Locust framework.
- Resource Utilization Metrics: The docker stats command was executed concurrently to collect time-series data on CPU and Memory usage for each container.

This data collection strategy enables a multi-faceted analysis of the architectural trade-offs, covering both system performance and resource efficiency.

6.3. Preliminary Empirical Results

This section presents the quantitative results of our comparative experiments. These results should be interpreted as a preliminary feasibility study illustrating the high-level characteristics of these different architectures.

It is critical to note that this prototype was not optimized for throughput; for example, it does not implement horizontal consumer scaling. As detailed in Section 7.3, the study's goal was to test architectural resilience (i.e., the system's failure mode under load), not to serve as a definitive performance benchmark. The following results should be interpreted in that context.

6.3.1. The Efficiency Champion: Validating the Cost of Decoupling

Our first analysis sought to empirically validate the theoretical expectation that distributed systems incur higher operational overhead (related to NQ1: Performance and Resource Efficiency). This directly answers RQ-AE1. The results from Figure 3 and Figure 4 confirm this hypothesis.

As shown in Figure 3, the BPC4MSA prototype consistently consumes significantly more CPU resources (40-43%) than its monolithic and SOA counterparts (23-28%). This demonstrates the tangible computational cost of the event-driven model's network communication, message serialization, and distributed coordination.

This finding is further reinforced by Figure 4, which measures the throughput achieved per unit of CPU cost. The Monolithic and SOA architectures are the "efficiency champions" in this test, peaking at over 10,000 RPS/CPU%. In stark contrast, the BPC4MSA prototype is approximately 40% less efficient, confirming it requires more resources to perform the same amount of work. This overhead can be considered a "resilience tax" for the decoupled design.

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 December 2025 doi:10.20944/preprints202512.0916.v1

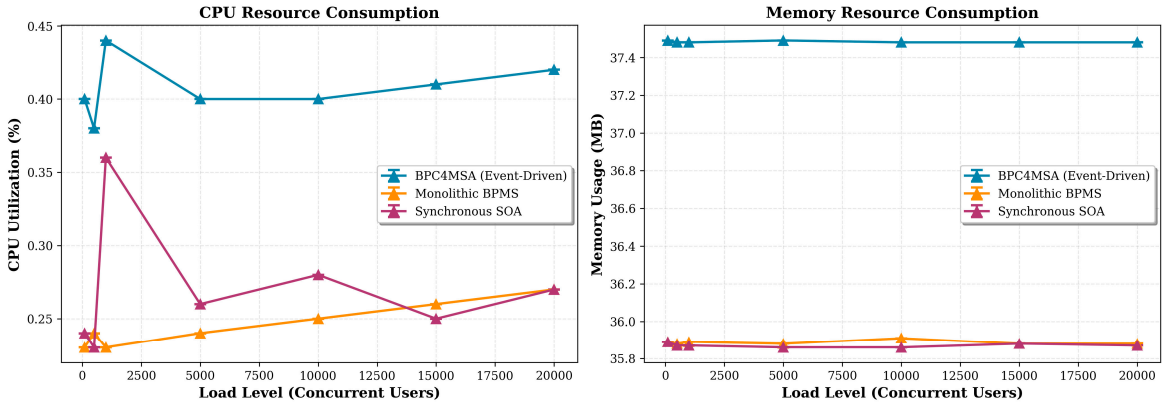

Figure 3. Resource Utilization (CPU and Memory).

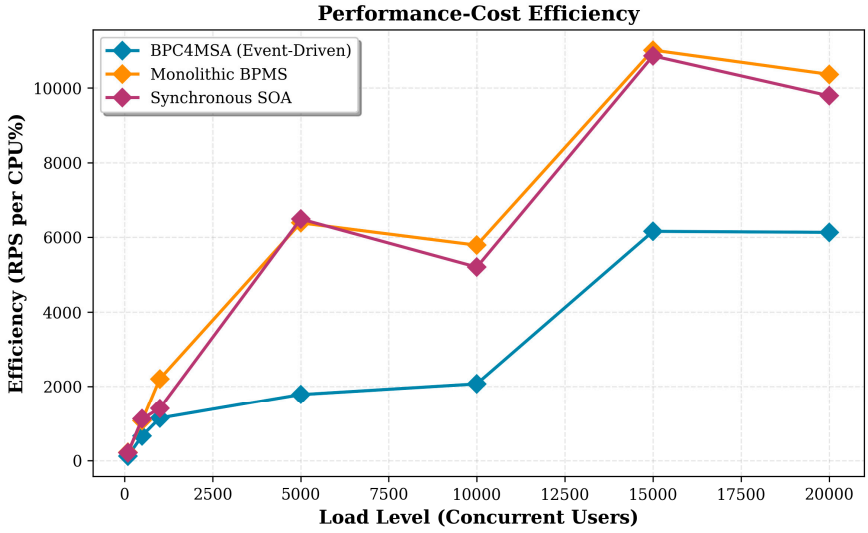

Figure 4. Performance-Cost Efficiency.

6.3.2. Comparative Analysis of System Stability

This section addresses RQ-AE2, which questions how the architectures differ in their stability and resilience. We identify the 'collapse' point as the load level where a sustained performance decline begins. As shown in Figure 5, the Monolithic and SOA architectures reach this inflection point at 15,000 users, with throughput declining sharply thereafter. This represents a brittle 'fast-fail' state, where the system becomes unavailable. In contrast, the event-driven BPC4MSA architecture avoids this throughput collapse, demonstrating the load-leveling capabilities of its asynchronous, buffered design.

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 December 2025 doi:10.20944/preprints202512.0916.v1

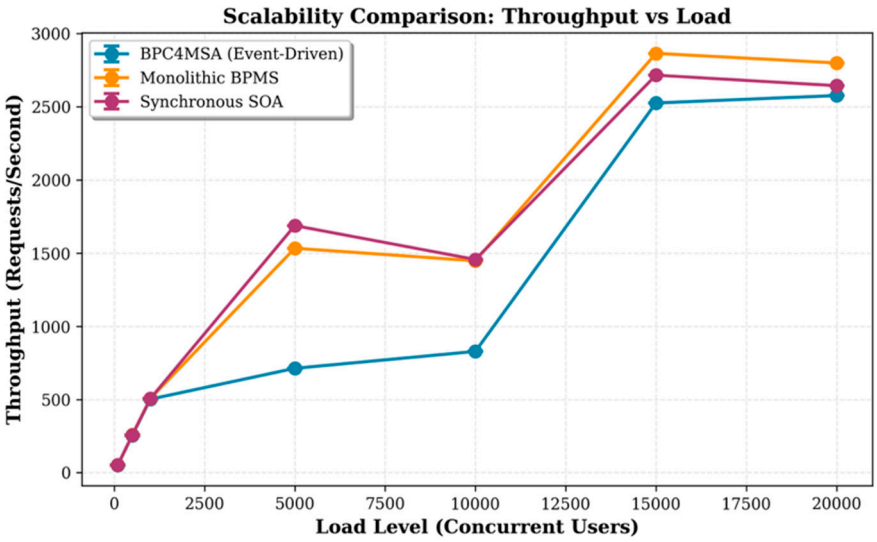

Figure 5. Scalability Comparison.

7. Discussion

7.1. Synthesis of Findings

Our dual-method evaluation provides a clear validation path for the BPC4MSA framework. The Design Evaluation (SLR in Section 5) established the framework's conceptual novelty and its comprehensive scope in addressing identified research gaps. The Prototype Evaluation (empirical study in Section 6) served as a preliminary feasibility check of its runtime component, confirming the expected behavioral trade-offs of its asynchronous, event-driven pattern. This discussion synthesizes these two findings, focusing first on the primary implications of the conceptual framework, and second, on the practical implementation insights from the prototype.

7.2. Implications of the Conceptual Framework

The primary contribution of this research is the BPC4MSA framework itself, a novel conceptual artifact. The SLR (Section 5) confirmed that its design is unique in holistically integrating solutions to persistent, systemic gaps (GAPs 1-5) that prior research has typically addressed in isolation.

The most significant conceptual implication is the framework's proposed mechanism for closing the "Broken Feedback Loop" (GAP-2). By architecting an integrated system where the Compliance Reporting Service (C4, instantiating f_{report}) and the Compliance Metric Formalization Service (C1, instantiating f_{form}) are connected via a common Event Bus, the design *theoretically* enables runtime compliance insights to programmatically inform and update design-time policies. While the empirical validation of this specific loop is a key area for future work, its inclusion in the *design* is a novel contribution that moves beyond static, siloed solutions.

Similarly, the framework's design addresses the "Last Mile Problem" (GAP-1) by dedicating specific services (Compliance Metric Formalization Service, Regulation Document Parsing Service) to the formalization of policy, creating a foundation for future semi-automated or AI-driven translation of legal text. For practitioners, this design offers a scalable, cloud-native blueprint for BPC that aligns with modern MSA/Cloud principles, prioritizing decoupling and lifecycle management over traditional, centralized, and brittle compliance tools.

7.3. Feasibility and Implementation Challenges of the Runtime Pattern

The secondary, empirical study (Section 6) was intentionally limited. It was not a comprehensive benchmark but a preliminary feasibility check of the runtime auditing pattern (C3), specifically instantiations of the f_{log} and f_{check} functions. The goal was to test its *behavior* under load, not its *optimized performance*.

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 December 2025 doi:10.20944/preprints202512.0916.v1

The findings from this prototype must be interpreted through the lens of its known limitations (Section 7.4). The experiment confirmed a known property of event-driven architectures: an un-scaled consumer creates a processing bottleneck. The "stability" observed in Figure 5 is not a novel discovery of "resilience," but a confirmation that the message queue successfully buffered the load, deferring failure by transforming a "fast-fail" (system crash) into a "slow-fail" (an infinitely growing processing queue).

The implication for practice is significant: while the BPC4MSA architecture is well-suited for high-load, asynchronous auditing where system survival and guaranteed ingestion is paramount, its implementation inherently involves complex engineering considerations. This study empirically confirms that consumer scaling is not an optional optimization but a mandatory design requirement to manage processing latency and deliver a functional system.

7.4. Limitations and Future Work

The primary limitation of this research is the delta between the holistic conceptual design and the preliminary empirical validation. Our study validated the conceptual novelty of the full framework (Section 5) but only tested the basic feasibility of one component (Section 6).

Future work must focus on closing this gap. The most critical next step is the implementation and empirical validation of the framework's most novel conceptual feature: the mechanism for closing the "Broken Feedback Loop" (GAP-2). This involves empirically testing the feedback path from the Compliance Reporting Service back to the Compliance Metric Formalization Service. A second major avenue is to address the "Last Mile Problem" (GAP-1) by exploring AI and NLP techniques within the dedicated formalization services. Finally, the runtime prototype itself must be advanced with proper consumer scaling to conduct realistic, end-to-end latency benchmarks.

8. Conclusion

This research addressed the significant challenge of ensuring Business Process Compliance within modern MSA and Cloud environments. Using a Design Science Research Methodology, we first identified critical research gaps—including a "Broken Feedback Loop"—and derived a set of requirements for a novel solution. We then designed and instantiated the BPC4MSA framework, a cloud-native, microservices-based architecture.

Our dual-method evaluation yielded two distinct contributions. The primary contribution is the BPC4MSA framework as a conceptually novel artifact. Our Design Evaluation (Section 5), a Systematic Literature Review, confirmed that the framework's design holistically addresses persistent gaps in the state-of-the-art, particularly by providing a theoretical mechanism to close the "Broken Feedback Loop".

The secondary contribution was a preliminary empirical feasibility evaluation of the framework's runtime component. This study confirmed the expected behavioral trade-offs of the event-driven pattern, validating its suitability for high-load asynchronous auditing (where system uptime takes precedence over immediate processing) while empirically demonstrating that effective realization—particularly through appropriate consumer scaling—is essential to manage processing latency.

Further research is required to empirically validate the framework's full-lifecycle capabilities, particularly its proposed mechanisms for addressing the "Last Mile Problem" and closing the "Broken Feedback Loop." However, the BPC4MSA framework provides a robust, validated foundation for achieving agile, scalable, and verifiable compliance in modern cloud architectures.

Author Contributions: Conceptualization, N. Long Ha and Thomas M. Prinz; Methodology, Formal Analysis, and Investigation, N. Long Ha and Thomas M. Prinz; Software, Validation, and Data Curation, N. Long Ha, S. Huong Do, and Quan Truong Tan; Writing—Original Draft, N. Long Ha; Writing—Review and Editing, N. Long Ha, S. Huong Do, Quan Truong Tan, and Thomas M. Prinz; Visualization, N. Long Ha; Supervision and Project

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 December 2025 doi:10.20944/preprints202512.0916.v1

Administration, N. Long Ha and Thomas M. Prinz. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Hue University, grant number DHH2023-06-135.

Data Availability Statement: The complete source code and experimental scripts used for the prototype evaluation described in Section 6 are publicly available at https://github.com/hangoclong/bpc4msa.git.

Acknowledgments: The authors gratefully acknowledge the administrative and technical support from the University of Economics, Hue University. This research was supported by Hue University under grant number DHH2023-06-135.

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

References

1. Dumas M, La Rosa M, Mendling J, Reijers HA (2018) Fundamentals of Business Process Management. Springer Berlin Heidelberg, Berlin, Heidelberg
2. El Kharbili M, Alves De Medeiros AK, Stein S, Van Der Aalst WMP (2008) Business process compliance checking: Current state and future challenges. In: Lecture Notes in Informatics (LNI), Proceedings - Series of the Gesellschaft fur Informatik (GI). pp 107–113
3. Governatori G (2005) Representing Business Contracts In RuleML. Int J Coop Inf Syst 14:181–216. https://doi.org/10.1142/S0218843005001092
4. Governatori G, Sadiq S (2008) The Journey to Business Process Compliance. Handbook of Research on Business Process Modeling 426–454. https://doi.org/10.4018/978-1-60566-288-6.ch020
5. Dragoni N, Giallorenzo S, Lafuente AL, Mazzara M, Montesi F, Mustafin R, Safina L (2017) Microservices: Yesterday, Today, and Tomorrow. In: Present and Ulterior Software Engineering. Springer International Publishing, Cham, pp 195–216
6. Mustapha AM, Arogundade O 'Tale, Abayomi-Alli A, Adeniran OJ, Adesemowo K, Alonge CY (2020) A Systematic Method for Extracting and Analyzing Cloud-Based Compliance Requirements. In: 2020 International Conference in Mathematics, Computer Engineering and Computer Science (ICMCECS). IEEE, pp 1–7
7. Yimam D, Fernandez EB (2016) A survey of compliance issues in cloud computing. Journal of Internet Services and Applications 7:5. https://doi.org/10.1186/s13174-016-0046-8
8. Hevner AR, March ST, Park J, Ram S (2004) Design Science in Information Systems Research. MIS Quarterly 28:75. https://doi.org/10.2307/25148625
9. Peffers K, Tuunanen T, Rothenberger MA, Chatterjee S (2007) A Design Science Research Methodology for Information Systems Research. Journal of Management Information Systems 24:45–77. https://doi.org/10.2753/MIS0742-1222240302
10. Sutcliffe A (2003) Scenario-based requirements engineering. In: Proceedings. 11th IEEE International Requirements Engineering Conference, 2003. pp 320–329
11. Polyvyanyy A, Ouyang C, Barros A, van der Aalst WMP (2017) Process querying: Enabling business intelligence through query-based process analytics. Decis Support Syst 100:41–56. https://doi.org/10.1016/j.dss.2017.04.011
12. Hashmi M, Governatori G, Lam HP, Wynn MT (2018) Are we done with business process compliance: state of the art and challenges ahead. Knowl Inf Syst 57:79–133. https://doi.org/10.1007/s10115-017-1142-1
13. Viriyasitavat W, Da Xu L, Dhiman G, Bi Z (2023) Blockchain-as-a-Service for Business Process Management: Survey and Challenges. IEEE Trans Serv Comput 16:2299–2314. https://doi.org/10.1109/TSC.2022.3199232
14. Elgammal A, Turetken O (2015) Lifecycle Business Process Compliance Management: A Semantically-Enabled Framework. In: 2015 International Conference on Cloud Computing, ICCC 2015. IEEE, pp 1–8

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 December 2025 doi:10.20944/preprints202512.0916.v1

15. Sackmann S, Kuehnel S, Seyffarth T (2018) Using Business Process Compliance Approaches for Compliance Management with Regard to Digitization: Evidence from a Systematic Literature Review. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Springer Verlag, pp 409–425
16. Kuehnel S, Sackmann S, Damarowsky J, Boehmer M (2022) EconBPC: A Tool for Activity-based Monetary Assessment and Visualization of Security and Compliance Measures in Business Processes
17. Fellmann M, Zasada A (2014) State-of-the-art of business process compliance approaches: A Survey. ECIS 2014 Proceedings - 22nd European Conference on Information Systems 1–17
18. Mustapha AM, Arogundade O 'Tale, Abayomi-Alli A, Adesemowo AK, Adeniran OJ (2024) An improved cloud-based business process compliance management system using a user-centered approach. International Journal of System Assurance Engineering and Management 15:5111–5138. https://doi.org/10.1007/s13198-024-02494-6
19. García-Galán J, Pasquale L, Grispos G, Nuseibeh B (2016) Towards adaptive compliance. In: Proceedings of the 11th International Symposium on Software Engineering for Adaptive and Self-Managing Systems. ACM, New York, NY, USA, pp 108–114
20. Awad A, Sakr S, Elgammal A (2015) Compliance Monitoring as a Service: Requirements, Architecture and Implementation. In: 2015 International Conference on Cloud Computing (ICCC). IEEE, pp 1–7
21. Orriens B, Heuvel WJ V.D., Papazoglou M (2008) On the risk management and auditing of SOA based business processes. In: Communications in Computer and Information Science. pp 124–138
22. Barnawi A, Awad A, Elgammal A, El Shawi R, Almalaise A, Sakr S (2015) Runtime self-monitoring approach of business process compliance in cloud environments. Cluster Comput 18:1503–1526. https://doi.org/10.1007/s10586-015-0494-0
23. Koetter F, Kochanowski M, Renner T, Fehling C, Leymann F (2013) Unifying Compliance Management in Adaptive Environments through Variability Descriptors (Short Paper). In: 2013 IEEE 6th International Conference on Service-Oriented Computing and Applications. IEEE, pp 214–219
24. Mustapha AM, Arogundade OT, Misra S, Damasevicius R, Maskeliunas R (2020) A systematic literature review on compliance requirements management of business processes. International Journal of System Assurance Engineering and Management 11:561–576. https://doi.org/10.1007/s13198-020-00985-w
25. Compagna L, Guilleminot P, Brucker AD (2013) Business Process Compliance via Security Validation as a Service. In: 2013 IEEE Sixth International Conference on Software Testing, Verification and Validation. IEEE, pp 455–462
26. Kratzke N, Quint PC (2017) Understanding cloud-native applications after 10 years of cloud computing - A systematic mapping study. Journal of Systems and Software 126:1–16. https://doi.org/10.1016/j.jss.2017.01.001
27. Richardson C (2018) Microservices patterns: with examples in Java. Simon and Schuster
28. Ortiz J, Torres V, Valderas P (2023) Microservice compositions based on the choreography of BPMN fragments: facing evolution issues. Computing 105:375–416. https://doi.org/10.1007/s00607-022-01128-8
29. Alpers S, Becker C, Oberweis A, Schuster T (2015) Microservice Based Tool Support for Business Process Modelling. In: 2015 IEEE 19th International Enterprise Distributed Object Computing Workshop. IEEE, pp 71–78
30. Borkowski M, Fdhila W, Nardelli M, Rinderle-Ma S, Schulte S (2018) Event-based failure prediction in distributed business processes. Inf Syst 0:1–17. https://doi.org/10.1016/j.is.2017.12.005
31. Singh Aujla G, Barati M, Rana O, Dustdar S, Noor A, Llanos JT, Carr M, Marikyan D, Papagiannidis S, Ranjan R (2020) COM-PACE: Compliance-Aware Cloud Application Engineering Using Blockchain. IEEE Internet Comput 24:45–53. https://doi.org/10.1109/MIC.2020.3014484
32. Casalicchio E, Cardellini V, Interino G, Palmirani M (2018) Research challenges in legal-rule and QoS-aware cloud service brokerage. Future Generation Computer Systems 78:211–223. https://doi.org/10.1016/j.future.2016.11.025
33. Fang Z, Yin C (2010) BPM Architecture Design Based on Cloud Computing. Intell Inf Manag 02:329–333. https://doi.org/10.4236/iim.2010.25039

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 December 2025 doi:10.20944/preprints202512.0916.v1

34. Rimol M (2021) Four Trends Are Shaping the Future of Public Cloud. In: Gartner. gartner.com/en/newsroom/press-releases/2021-08-02-gartner-says-four-trends-are-shaping-the-future-of-public-cloud. Accessed 21 Mar 2022
35. Cloud Native Computing Foundation (2015) Mission of the Cloud Native Computing Foundation. https://github.com/cncf/foundation/blob/main/charter.md. Accessed 21 Mar 2022
36. Accorsi R, Lowis L, Sato Y (2011) Automated certification for compliant cloud-based business processes. Business and Information Systems Engineering 3:145–154. https://doi.org/10.1007/s12599-011-0155-7
37. Brucker AD, Compagna L, Guilleminot P (2014) Compliance Validation of Secure Service Compositions. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). pp 136–149
38. Birukou A, Cavalcante AB, Casati F, Chowdhury SR, D'Andrea V, Leymann F, Oberortner E, Serafinski J, Silveira P, Strauch S, Tluczek M (2010) An integrated solution for runtime compliance governance in SOA. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). pp 706–707
39. Tran H, Holmes T, Oberortner E, Mulo E, Cavalcante AB, Serafinski J, Tluczek M, Birukou A, Daniel F, Silveira P, Zdun U, Dustdar S (2010) An End-to-End Framework for Business Compliance in Process-Driven SOAs. In: 2010 12th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing. IEEE, pp 407–414
40. Vetter A (2016) Detecting Operator Errors in Cloud Maintenance Operations. In: 2016 IEEE International Conference on Cloud Computing Technology and Science (CloudCom). IEEE, pp 639–644
41. Indiono C, Fdhila W, Rinderle-Ma S (2018) Evolution of Instance-Spanning Constraints in Process Aware Information Systems. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Springer Verlag, pp 298–317
42. Seyffarth T, Kuehnel S, Sackmann S (2018) Business process compliance and business process change: An approach to analyze the interactions. In: Lecture Notes in Business Information Processing. Springer Verlag, pp 176–189
43. Koetter F, Kintz M, Kochanowski M, Fehling C, Gildein P, Leymann F, Weisbecker A (2016) Unified Compliance Modeling and Management using Compliance Descriptors. In: Proceedings of the 6th International Conference on Cloud Computing and Services Science. SCITEPRESS - Science and Technology Publications, pp 159–170
44. Koetter F, Kintz M, Kochanowski M, Wiriyarattanakul T, Fehling C, Gildein P, Wagner S, Leymann F, Weisbecker A (2017) An Universal Approach for Compliance Management Using Compliance Descriptors. In: Helfert M, Ferguson D, Méndez Muñoz V, Cardoso J (eds) Communcations in Computer and Information Science. Springer International Publishing, Cham, pp 209–231
45. Filepp R, Adam C, Hernandez M, Vukovic M, Anerousis N, Zhang GQ (2018) Continuous compliance: Experiences, challenges, and opportunities. In: Proceedings - 2018 IEEE World Congress on Services, SERVICES 2018. Institute of Electrical and Electronics Engineers Inc., pp 21–22
46. Kuehnel S, Zasada A (2018) An Approach Toward the Economic Assessment of Business Process Compliance. In: Woo C, Lu J, Li Z, Ling TW, Li G, Lee ML (eds). Springer International Publishing, Cham, pp 228–238
47. Loreti D, Chesani F, Ciampolini A, Mello P (2018) A distributed approach to compliance monitoring of business process event streams. Future Generation Computer Systems 82:104–118. https://doi.org/10.1016/j.future.2017.12.043
48. Adams N, Augusto A, Davern M, Rosa M La (2022) Why Do Banks Find Business Process Compliance so Challenging? An Australian Perspective. In: Lecture Notes in Business Information Processing. pp 3–20
49. Ly LT, Göser K, Rinderle-Ma S, Dadam P (2008) Compliance of semantic constraints - A requirements analysis for process management systems. In: CEUR Workshop Proceedings. pp 31–45
50. Scholte T, Kirda E (2010) Achieving Life-Cycle Compliance of Service-Oriented Architectures: Open Issues and Challenges. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). pp 253–267

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 December 2025 doi:10.20944/preprints202512.0916.v1

51. Mustapha AM, Arogundade OT, Vincent OR, Adeniran OJ, Chen X (2017) A model-based business process compliance management architecture for SMSE towards effective adoption of cloud computing. In: 2017 International Conference on Computing Networking and Informatics (ICCNI). IEEE, pp 1–6
52. Ouyang R, Wang J, Xu H, Chen S, Xiong X, Tolba A, Zhang X (2023) A Microservice and Serverless Architecture for Secure IoT System. Sensors 23:4868. https://doi.org/10.3390/s23104868
53. Suliman AT, Kadadha M, Mizouni R, Otrok H, Damiani E, Al-Qutayri M (2023) Blockcheck: A consortium blockchain-based conformance checking framework for business processes. Internet of Things 21:100652. https://doi.org/10.1016/j.iot.2022.100652
54. Singh P (2022) Designing Observable Microservices for Financial Applications with Built-in Compliance. International Journal of Multidisciplinary Research and Growth Evaluation 3:1163–1168. https://doi.org/10.54660/.IJMRGE.2022.3.1.1163-1168
55. Rodríguez C, Schleicher D, Daniel F, Casati F, Leymann F, Wagner S (2013) SOA-enabled compliance management: instrumenting, assessing, and analyzing service-based business processes. Service Oriented Computing and Applications 7:275–292. https://doi.org/10.1007/s11761-013-0129-3
56. Kellogg M, Schäf M, Tasiran S, Ernst MD (2020) Continuous compliance. In: Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering. ACM, New York, NY, USA, pp 511–523
57. Lu R, Sadiq S, Governatori G (2008) Compliance aware business process design. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). pp 120–131
58. Tsakalidis G, Vergidis K, Delias P, Vlachopoulou M (2019) A Conceptual Business Process Entity with Lifecycle and Compliance Alignment. In: Lecture Notes in Business Information Processing. pp 70–82
59. Beckers K (2015) Supporting the Establishment of a Cloud-Specific ISMS According to ISO 27001 Using the Cloud System Analysis Pattern. In: Pattern and Security Requirements. Springer International Publishing, Cham, pp 299–392
60. Silveira P, Rodríguez C, Birukou A, Casati F, Daniel F, D'Andrea V, Worledge C, Taheri Z (2012) Aiding Compliance Governance in Service-Based Business Processes. In: Handbook of Research on Service-Oriented Systems and Non-Functional Properties. IGI Global, pp 524–548
61. Silveira P, Rodríguez C, Casati F, Daniel F, D'Andrea V, Worledge C, Taheri Z (2010) On the design of compliance governance dashboards for effective compliance and audit management. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). pp 208–217
62. Adams N, Augusto A, Davern M, La Rosa M (2024) End-to-end, no-code business process compliance framework for the banking industry. In: CEUR Workshop Proceedings. pp 15–27
63. Namiri K, Stojanovic N (2007) Pattern-based design and validation of business process compliance. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). pp 59–76
64. Yao J, Chen S, Levy D (2014) Accountability-Based Compliance Control of Collaborative Business Processes in Cloud Systems. In: Security, Privacy and Trust in Cloud Systems. Springer Berlin Heidelberg, Berlin, Heidelberg, pp 345–374
65. Kiesel J, Grünewald E (2024) Extending Business Process Management for Regulatory Transparency. In: Business Process Management Forum. pp 337–353
66. Vadlamudi S, Sam J (2021) A Novel Approach to Onboarding Secure Cloud-Native Acquisitions into Enterprise Solutions. In: 2021 International Conference on Disruptive Technologies for Multi-Disciplinary Research and Applications (CENTCON). IEEE, pp 228–233
67. Monteiro D, Maia PHM, Rocha LS, Mendonça NC (2020) Building orchestrated microservice systems using declarative business processes. Service Oriented Computing and Applications 14:243–268. https://doi.org/10.1007/s11761-020-00300-2
68. Zarour K, Benmerzoug D (2018) Multicriteria-Based Analysis and Evaluation of Business Processes Executed in Multi-Cloud Environment. In: Advances in Intelligent Systems and Computing. Springer Verlag, pp 315–327

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 December 2025 doi:10.20944/preprints202512.0916.v1

69. Casalicchio E, Palmirani M (2015) A Cloud Service Broker with Legal-Rule Compliance Checking and Quality Assurance Capabilities
70. Solaiman E, Sfyrakis I, Molina-Jimenez C (2016) A State Aware Model and Architecture for the Monitoring and Enforcement of Electronic Contracts. In: Proceedings - CBI 2016: 18th IEEE Conference on Business Informatics. Institute of Electrical and Electronics Engineers Inc., pp 55–63
71. Accorsi R (2011) Business process as a service: Chances for remote auditing. In: Proceedings - International Computer Software and Applications Conference. pp 398–403
72. Garg R, Naudts B, Verbrugge S, Stiller B (2015) Modeling legal and regulative requirements for ranking alternatives of cloud-based services. 8th International Workshop on Requirements Engineering and Law, RELAW 2015 - Proceedings 25–32. https://doi.org/10.1109/RELAW.2015.7330208
73. Liu X, Xia C, Wang T, Zhong L (2017) CloudSec: A Novel Approach to Verifying Security Conformance at the Bottom of the Cloud. In: 2017 IEEE International Congress on Big Data (BigData Congress). IEEE, pp 569–576
74. Mulo E, Zdun U, Dustdar S (2013) Domain-specific language for event-based compliance monitoring in process-driven SOAs. Service Oriented Computing and Applications 7:59–73. https://doi.org/10.1007/s11761-012-0121-3
75. Seyffarth T, Kuehnel S (2022) Maintaining business process compliance despite changes: a decision support approach based on process adaptations. J Decis Syst 31:305–335. https://doi.org/10.1080/12460125.2020.1861920
76. Seyffarth T, Raschke K (2020) BCIT: A tool to recommend compliant business processes based on process adaption
77. Stratigaki C, Nikolaidou M, Loucopoulos P, Anagnostopoulos D (2016) Business process elicitation from regulatory compliance documents: An E-government case study
78. Teixeira Filho HM, Guerreiro Azevedo L, Matsui Siqueira SW (2016) IntelliGOV - A semantic approach for compliance validation of service-oriented architectures
79. Vetter A (2019) Online Detection of Operator Errors in Cloud Computing Using Anti-patterns. In: Lecture Notes in Business Information Processing. pp 1–24
80. Tosatto SC, Governatori G, Kelsen P (2015) Business Process Regulatory Compliance is Hard. IEEE Trans Serv Comput 8:958–970. https://doi.org/10.1109/TSC.2014.2341236

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

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.