systems

MDPI

Article

Towards Governance of Socio-Technical System of Systems: Leveraging Lessons from Proven Engineering Principles

Mohamed Mogahed * and Mo Mansouri

Department of Systems Engineering, Stevens Institute of Technology, Hoboken, NJ 07030, USA; mmansour@stevens.edu
* Correspondence: mmogahed@stevens.edu

Abstract

Healthcare delivery systems operate as complex socio-technical Systems-of-Systems (SoS), where autonomous entities—hospitals, insurers, laboratories, and technology vendors—must coordinate to achieve collective outcomes that exceed individual capabilities. Despite substantial investment in interoperability standards and regulatory frameworks, persistent fragmentation undermines care quality, operational efficiency, and systemic adaptability. This fragmentation stems from a fundamental governance paradox: how can independent systems retain operational autonomy while adhering to shared rules that ensure systemic resilience? This paper addresses this challenge by advancing a governance-oriented architecture grounded in Object-Oriented Programming (OOP) principles. We reinterpret core OOP constructs—encapsulation, modularity, inheritance, polymorphism, and interface definition—as governance mechanisms that enable autonomy through principled constraints while fostering structured coordination across heterogeneous systems. Central to this framework is the Confluence Interoperability Covenant (CIC), a socio-technical governance artifact that functions as an adaptive interface mechanism, codifying integrated legal, procedural, and technical standards without dictating internal system architectures. To validate this approach, we develop a functional proof-of-concept simulation using Petri Nets, modeling constituent healthcare systems as autonomous entities interacting through CIC-governed transitions. Comparative simulation results demonstrate that CIC-based governance significantly reduces fragmentation (from 0.8077 to 0.1538) while increasing successful interactions fivefold (from 68 to 339 over 400 steps). This work contributes foundational principles for SoS Engineering and offers practical guidance for designing scalable, interoperable governance architectures in mission-critical socio-technical domains.

Keywords: systems of systems; socio-technical; healthcare; governance; object-oriented programming; autonomy; confluence interoperability covenant; interoperability; fragmentation; policy

check for updates

Academic Editors: Polinpapilinho Katina and Chuck Keating

Received: 26 September 2025
Revised: 4 November 2025
Accepted: 5 December 2025
Published: 10 December 2025

Citation: Mogahed, M.; Mansouri, M. Towards Governance of Socio-Technical System of Systems: Leveraging Lessons from Proven Engineering Principles. *Systems* **2025**, *13*, 1113. https://doi.org/10.3390/systems13121113

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Healthcare delivery today functions as a sprawling socio-technical System-of-Systems (SoS), comprising autonomous hospitals, clinics, pharmacies, insurers, IT vendors, and regulators that must coordinate to deliver safe, effective, and timely care [1]. Yet, despite substantial investments in interoperability standards and compliance regimes, the ecosystem remains fragmented: disparate data formats, ad hoc integrations, misaligned incentives, and conflicting procedural norms inhibit information flow and stifle innovation [2]. As a

result, patient outcomes suffer, operational costs rise, and the system's ability to adapt to emerging challenges—such as pandemics or novel care pathways—is undermined [3].

Most existing governance approaches address this fragmentation in silos. Technical standards (e.g., HL7, FHIR) define message structures but lack binding procedural or legal context; legal agreements (e.g., business associate agreements, service contracts) impose high-level obligations yet rarely enforce runtime conformance; institutional policies (e.g., consent protocols, accreditation rules) fill procedural gaps but often conflict or overlap [4]. The net effect is a patchwork of instruments that constituent systems must interpret separately, leading to brittle integrations and governance overhead [5].

At its core, this challenge mirrors a paradox: how can independent systems retain operational independence while adhering to shared rules that ensure systemic resilience? To address this fundamental question, Baxter and Sommerville (2011) highlight the balance between control and autonomy, arguing that loosely coupled coordination mechanisms—such as flexible standards and interface agreements—allow independent systems to adapt while still conforming to overarching objectives [6]. Behymer and Flach (2017) propose a systems design perspective that emphasizes collaborative affordances, suggesting that rather than reducing autonomy, system-level resilience can be achieved by enhancing the capacity of autonomous agents to perceive, understand, and act on shared constraints [7]. Cardoso et al. (2022) focus on the role of normative frameworks and runtime monitoring in multi-agent systems, where agents maintain local autonomy but are incentivized or constrained by system-level rules that promote reliability and coordination [8]. In a related vein, Sawaragi (2020) emphasizes co-creation between humans and systems to embed resilience principles within both organizational routines and technical architectures, thereby aligning individual system goals with emergent systemic behavior [9]. Finally, Heo and Yi (2023) suggested that decentralized, contract-based governance mechanisms can encode shared expectations and accountability without imposing centralized control, enabling a distributed yet coherent form of resilience [10].

While existing research has recognized the persistent tension between autonomy and centralized control in socio-technical SoS, few studies have effectively translated established engineering principles into operational governance models capable of bridging this divide. This paper advances both the principles of SoS Engineering and the practice of SoS governance by leveraging proven systems design concepts to propose a governance-oriented architecture specifically tailored for socio-technical contexts. Building on the conceptual foundation of Object-Oriented Programming (OOP), the paper reinterprets its key constructs—encapsulation, modularity, inheritance, and interface definition—as governance mechanisms that enable autonomy through principled constraints while fostering structured coordination across diverse systems. In this sense, the framework does not merely borrow from software design but extends its logic to advance foundational principles for designing, analyzing, and governing complex SoS. Just as OOP enables scalable and resilient software through clear boundaries and interaction contracts, the proposed paradigm provides a transferable blueprint for engineering scalable, adaptive, and interoperable SoS such as healthcare. The discussion that follows articulates the broader implications of this translation, demonstrating how it informs SoS Engineering principles, meta-governance design, and systemic resilience strategies, offering both academic insight and practical benefit for the governance of distributed, mission-critical systems.

To validate the proposed governance construct, we develop a formal simulation using Petri Nets, enabling computational assessment of its structural and behavioral implications. In this model, constituent systems such as hospitals, insurers, and laboratories are represented as autonomous entities interacting through defined transitions governed by the Confluence Interoperability Covenant (CIC), a formal coordination mechanism that

encodes shared rules, interaction protocols, and compliance expectations. This modeling approach not only demonstrates the feasibility of operationalizing governance principles within a system-of-systems architecture but also provides a quantitative basis for evaluating their effectiveness. Simulation results reveal that CIC governance enhances adaptive coordination across heterogeneous healthcare actors through polymorphic interaction pathways while reducing fragmentation and reinforcing systemic cohesion. Furthermore, the findings illuminate the dynamic nature of compliance and oversight, underscoring the need for continuous policy adaptation as a critical implication for real-world governance practice. The computational validation substantiates the paper's theoretical proposition and extends the discourse on SoS Engineering and governance design, highlighting the potential of structured, interface-based mechanisms to achieve resilient socio-technical ecosystems.

The remainder of this paper is organized as follows. Section 2 reviews related work, highlighting the characterization of healthcare delivery systems (HDS) as a SoS and outlining the conceptual foundations of OOP that inform the proposed governance construct. Section 3 introduces the OOP-based governance analogy and establishes the conceptual linkage between engineering design principles and socio-technical governance mechanisms, forming the theoretical core of the paper. Section 4 presents the simulation developed to validate the feasibility and performance of the proposed construct, demonstrating how the application of structured governance interfaces can reduce fragmentation and enhance adaptive coordination among independent systems. Section 5 provides a comprehensive discussion of theoretical and practical implications, connecting the paper's outcomes to the advancement of SoS Engineering principles and the design of resilient governance architectures. Finally, Section 6 concludes the paper by synthesizing the main contributions, emphasizing the academic and practical benefits of the proposed framework, and outlining directions for future research toward more dynamic, scalable, and data-informed governance of socio-technical SoS.

2. Related Work

This section provides the foundational background across three critical domains that underpin our research approach. We examine the conceptualization of healthcare delivery systems as complex socio-technical SoS, explore governance mechanisms specifically designed for SoS environments, and investigate the OOP paradigm as a structural blueprint that offers transferable logic for managing complex system interactions through modular design and standardized protocols.

2.1. The Healthcare System as a Socio-Technical SoS

Healthcare delivery systems exemplify complex socio-technical Systems-of-Systems (SoS), where autonomous entities—hospitals, insurers, patients, and technology platforms—interact dynamically within a web of legal, procedural, and technical interdependencies [11]. These systems are intrinsically loosely coupled and characterized by emergent behavior, meaning that global system properties arise not from centralized design but from the collective dynamics of independent actors [12].

To that end, many studies studied it as a socio-technical SoS. Sittig and Singh (2010) introduced an eight-dimensional socio-technical model tailored to health information technology (HIT), highlighting how systemic performance depends on interactions among hardware, software, clinical workflows, regulatory environments, and user behavior [13]. Building on this, Carayon (2012) proposed the SEIPS (Systems Engineering Initiative for Patient Safety) model, which situates patient safety and care quality at the intersection of people, technologies, tasks, and environments—further reinforcing the framing of healthcare as a socio-technical SoS [14]. From a governance perspective, Dong, Sahu, and Black (2024)

examined Ontario's transition toward integrated healthcare, emphasizing the difficulties of coordinating across loosely coupled institutions with distinct policies, infrastructures, and incentives [15]. Complementary to this, Bennaceur et al. (2023) approached resilience in community healthcare systems through a socio-technical lens, arguing that sustained system performance under stress hinges on adaptive coordination mechanisms [16]. Finally, global policy-focused research by de Savigny and Adam (2009) and the WHO highlighted that achieving effective health system governance requires managing interdependencies across regulatory, technological, and institutional actors without sacrificing local autonomy [17]. Collectively, these studies establish a growing consensus that healthcare systems exhibit the defining characteristics of socio-technical SoS, thereby demanding governance frameworks that are adaptive, integrative, and systemically aware.

In a related context, recent advances in artificial intelligence have explored the use of deep learning architectures to enhance clinical decision-making and system-level diagnostics, illustrating the growing reliance on intelligent components within healthcare SoS. For example, the SVMVGGNet-16 model integrates Support Vector Machines with a VGGNet-16 feature extractor to improve lung cancer detection accuracy from CT scans, achieving significant gains in precision and recall through hybrid feature learning [18]. Similarly, the Multi-Scale Parallel CNN (MSP-CNN) has been applied to COVID-19 chest X-ray image denoising, effectively reducing noise while preserving diagnostic detail [19]. These studies demonstrate how autonomous, data-driven subsystems can augment the broader healthcare ecosystem by improving reliability and adaptive capability, further emphasizing the need for coherent governance structures that ensure their integration and coordination at scale.

This complexity intensifies the need for governance frameworks that can balance operational autonomy with systemic resilience [20].

Through systematic analysis of the HDS's actual operational characteristics, Mogahed and Mansouri characterized it as a representative case of a Collaborative SoS, wherein autonomous entities including hospitals, clinics, and insurance providers engage in voluntary cooperation to achieve collective care delivery capabilities that surpass the capacity of any individual component [21]. They contended that this collaborative arrangement is a central cause of the system's ongoing fragmentation. Because these entities operate autonomously and collaborate by choice rather than mandate, the system tends to exhibit behaviors such as decisions driven by self-interest, limited overall awareness, conflicting goals, protectionism over proprietary knowledge, and unaddressed systemic risks. Despite being commonly referred to as a unified healthcare system, it was never intentionally designed or structured as one and has never functioned as an integrated whole. Instead, it consists of loosely affiliated components that interact mainly out of practical necessity, not through a shared, system-wide strategy. Furthermore, they brought essential focus to the role of interdependence as a crucial factor influencing system performance. They highlighted that independently operated systems often form interdependent relationships to achieve outcomes that would be impossible for any single system to accomplish alone. This interdependence fosters cooperative dynamics—such as symbiotic and commensalistic relationships—that demonstrate mutual reliance and shared advantages through coordinated efforts. Within this framework, they argued that earlier studies placed too much emphasis on the autonomy of individual systems in terms of management and operation, while neglecting the structural and strategic benefits that arise from their often-overlooked interconnections.

These systems are built to support human endeavors, with the aim not only to understand their internal workings but also to actively influence and direct their performance [22]. To that end, the authors extended their theoretical contributions by offering a detailed

examination of established SoS structural typologies [23]. Using illustrative examples, they highlighted the limitations and challenges involved in reconfiguring the Collaborative Healthcare SoS into alternative forms. Their analysis revealed that such transitions are often hindered by fundamental tensions—such as value conflicts, mismatched requirements, and the ongoing struggle to preserve the autonomy of individual systems while pursuing collective SoS objectives. In response, they advocate for a shift in the engineering approach to SoS: the introduction of an external governing body responsible for managing and strategically harnessing interdependencies across constituent systems to promote alignment between emergent system behavior and higher-level goals as illustrated in Figure 1.

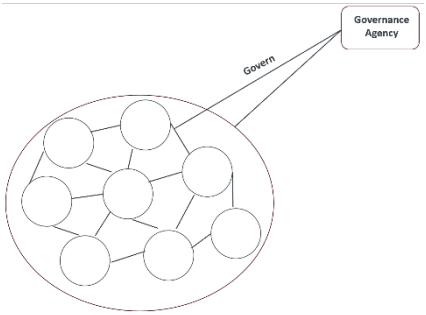


Figure 1. Proposed Construct of the Governing Entity of the SoS. Adopted from [23].

To anchor their structural assertions in governance theory, Mogahed and Mansouri conducted a systematic review of 45 studies covering 37 different governance frameworks [24]. This analysis identified fundamental governance mechanisms—such as accountability systems, feedback processes, and coordination methods—that are vital for enabling constituent systems to work together efficiently and accomplish objectives beyond their individual capabilities. Building on established governance frameworks such as the Systems-based Conceptual Model [25,26], the Complex System Governance (CSG) framework [27], and the combined perspectives of CSG and the Complex System Contextual Framework (CSCF) [28], the authors developed an enhanced System-of-Systems (SoS) governance model that delineates three core components: meta-governance, governed entities, and governing entities. This model provides a clearer structural perspective and offers a high-level abstraction capturing the key elements necessary for effective governance in diverse Collaborative Systems of Systems, including healthcare.

To validate the proposed model, the authors examined the mission of the U.S. Department of Health and Human Services (HHS) and developed both a functional categorization and an operational view of constituent systems according to their respective roles. They subsequently employed SysML modeling to represent the structural and operational dimensions of the healthcare SoS [29], as illustrated in Figure 2. Building on this foundation, graph theory and network analysis were applied to quantitatively assess the structural implications of introducing an external governance node into the healthcare SoS. The analysis revealed that the addition of a governance node substantially redistributed network influence by enhancing the centrality of providers and care recipients, while simultaneously reconfiguring the role of administrative and technological systems under a more coordinated oversight. This demonstrates the capacity of governance to steer system interactions toward alignment with overarching objectives. Parallel developments in graph-based learning have also advanced the modeling of complex, interconnected entities—a capability central to representing and analyzing SoS. The Pair-wise Hyper Graph Convolutional Network (PH-GCN), for instance, captures multi-level granularity and relational dependencies in human action recognition tasks, achieving superior robustness in complex motion scenarios [30]. Likewise, an Improved Graph Convolutional Network with Enriched

Graph Topology Representation enhances spatial–temporal feature integration for skeleton-based recognition [31]. Beyond their domain applications, these studies illustrate how graph-based architectures encapsulate structured interdependencies among autonomous agents, offering conceptual parallels to the coordination logic underpinning SoS governance. Such approaches underscore the increasing convergence between machine learning architectures and systems engineering models in addressing complexity, autonomy, and interaction dynamics.

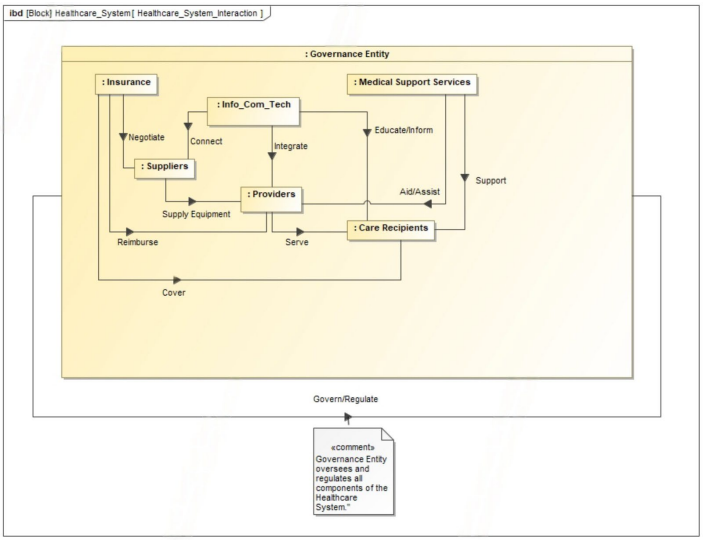


Figure 2. Internal Block Diagram (IBD): Operational Interactions Among Healthcare System Categories Under Governance Oversight. Adopted from [29].

To further clarify the governance entity within the proposed model, the authors elaborate on its structural composition, authority, and operational mechanisms, while also outlining the specific roles assigned to each component of the governance framework. The study systematically introduces a three-tiered governance architecture to illustrate how structured oversight can enhance coordination, accountability, and adaptability within the healthcare SoS [32]. At the highest level, Meta-Governance is fulfilled by the federal government, which establishes the overarching vision, sets national policy directives, and defines strategic priorities that guide the healthcare SoS toward collective objectives. This level provides the normative and strategic foundation for system-wide coherence, analogous to President John F. Kennedy's leadership during the Apollo program, where a clearly articulated national vision for space exploration galvanized alignment across multiple systems, institutions, and disciplines. Such meta-governance ensures a unifying direction that integrates political intent with systemic capability. The second tier, the Governing System, is operationalized through the HHS, which translates the federal vision into actionable governance mechanisms. It coordinates inter-system activities, aligns regulatory and operational processes, and enforces compliance with established policies. To ground this tier in institutional reality, the study conducts a detailed mapping of all HHS offices and agencies to their corresponding governance roles, demonstrating how distributed oversight can function cohesively through structured responsibilities and coordination interfaces. Finally, the Governed System encompasses the broader healthcare SoS, including providers, insurers, suppliers, and ancillary service entities that operate within the boundaries, standards, and feedback mechanisms defined by the higher tiers. This level represents the dynamic operational layer where governance is enacted in practice—where compliance, adaptation, and collaboration directly influence care delivery and system performance.

Moreover, the discussion emphasizes the interplay between governance and sustainability imperatives [33]. By embedding governance actions within the broader societal vision articulated by the United Nations' Sustainable Development Goal 3 (SDG 3)—to ensure healthy lives and promote well-being for all—the framework aligns institutional governance with global sustainability objectives. This integration reinforces that effective meta-governance not only coordinates technical and institutional systems but also anchors them to enduring social value, ensuring that healthcare evolution contributes to both systemic efficiency and collective human advancement.

While the review discussed in [24] offered valuable insights into SoS governance, there remains a critical need for methods that can formalize and validate the structural elements of such systems. As Falegnami et al. (2024) note in the context of socio-technical systems, conventional engineering approaches often fall short, highlighting the necessity for a paradigmatic shift in how these complex systems are conceptualized [22]. In response, alternative approaches advocate for reimagining socio-technical systems through new analytical lenses, drawing upon established principles from other engineering domains. Accordingly, this study turns to the foundational constructs of Object-Oriented Programming (OOP) as a source of conceptual inspiration, offering a structured and modular framework well-suited to modeling the dynamic roles, interactions, and governance mechanisms inherent in socio-technical SoS—particularly within the healthcare domain.

2.2. The Object-Oriented Blueprint

The origins of Object-Oriented Programming (OOP) lie in the 1960s, emerging from a broader trajectory of increasing abstraction in programming languages. Early computing required programmers to interact directly with memory addresses and hardware constraints, but the introduction of high-level languages opened the door for new metaphors such as objects, which could encapsulate both data and behavior. Simula, developed by Ole-Johan Dahl and Kristen Nygaard (1978) at the Norwegian Computing Center, is widely recognized as the first language to introduce the concept of objects [34]. Initially intended for simulation, Simula 67 formalized the idea of data structures with associated procedures—what would become objects—and introduced inheritance as a means to generalize class-based behaviors [35].

Around the same time, Ivan Sutherland's Sketchpad (2003) project laid a visual foundation for objects. Though not a programming language in the traditional sense, Sketchpad allowed for graphical constraints and the use of reusable components [36]. Its master-instance model and topological constraint mechanisms were early manifestations of encapsulated, self-contained entities in software. Nevertheless, the real conceptual leap came with Smalltalk, pioneered by Alan Kay and colleagues at Xerox PARC [37]. The paper traced its lineage from the initial prototype (Smalltalk-71) through Smalltalk-72, Smalltalk-76 and the first publicly released version, Smalltalk-80. Smalltalk introduced a purist form of object-orientation where everything—from numbers to control structures—was an object, and all computation occurred through message passing [38]. Smalltalk's architecture enforced a unified object model and introduced features like late binding, reflective metaclasses, and a graphical programming environment tailored for children and learners [39]. Later languages like C++, Java, and Python adopted OOP features, although often blending them with procedural paradigms, thus diluting the philosophical purity championed by Smalltalk [40].

The central principles of OOP, encapsulation, inheritance, and polymorphism, have remained relatively stable since Simula and Smalltalk.

Encapsulation involves bundling data with methods that operate on that data, and restricting direct access to some components of the object. This "black box" approach

protects the object's internal state from unintended interference, enabling modularity and robustness.

Inheritance enables a class to inherit properties and methods from another class, facilitating code reuse and the modeling of hierarchical relationships. This mechanism also supports the creation of abstract generalizations and specific implementations in object hierarchies.

Polymorphism allows objects to respond to the same method call in different ways depending on their type, a feature crucial for interface-based programming and dynamic behavior. This was formalized in theoretical models such as Abadi and Cardelli's object calculus.

These principles have allowed OOP to thrive as a design paradigm in both academic and industrial contexts. They also give rise to tradeoffs: while inheritance promotes reuse, overuse or deep hierarchies can lead to rigidity and complexity [41,42].

At its core, OOP represents a metaphor for modeling computation through entities (objects) that interact by sending messages. In Smalltalk, for instance, even classes are themselves objects—leading to a recursive ontology where objects instantiate from other objects and even define their own structure through metaclasses [37,39]. Objects are conceptualized as autonomous agents: they maintain internal state, expose behavior through interfaces, and interact without global oversight. This "object-as-agent" metaphor aligns OOP closely with actor models in concurrent systems [43], and even with philosophical constructs such as Leibnizian monads [37].

As Joque (2016) notes, the conceptual evolution of OOP also reflects philosophical tensions between linguistic representation and material computation [44]. In his view, the object in OOP is not merely a static data structure but a linguistic construction shaped by software metaphors and abstractions. Objects do not preexist in nature but emerge through programming practices as "negotiated and unstable means of counting the computational real". This view challenges the reductionist view of OOP as simply a technical methodology. Instead, it positions object-oriented thinking as a framework for engaging with the complexity of digital systems, where abstraction, delegation, and interaction serve as core structural devices [45,46].

In the same conceptual context, recent studies have revisited and expanded the conceptual structure of OOP, refining its theoretical underpinnings and adapting its principles to contemporary computational needs. Carvajal et al. (2018) provided a formal comparison between nominal and structural typing through the NOOP model, highlighting how real-world languages like Java and C++ rely on nominal inheritance structures to manage object identity, subtyping, and behavior [47]. Their work clarified foundational distinctions often blurred in traditional structural interpretations of OOP. Complementing this, the COMMOOP project (2016) offered a multidimensional framework for conceptualizing OOP learning and competence, identifying key cognitive and structural elements such as class-object relationships, abstraction, and algorithmic thinking [48]. Expanding further, Li et al. (2024) explored the continued relevance of OOP in AI and big data systems, demonstrating how encapsulation and polymorphism remain essential for maintaining modularity and scalability in machine learning pipelines [49]. Mars (2025) introduced the notion of "data spatial programming," extending the conceptual reach of OOP by embedding spatial and topological semantics into class-like constructs called archetypes—highlighting a shift toward modeling systems with emergent, distributed behaviors [50]. Together, these studies not only reinforce the enduring relevance of core OOP principles but also serve as a proven systems engineering blueprint for managing complexity through encapsulation of state, polymorphic interaction, and hierarchical abstraction.

In this paper, drawing from its foundational constructs, we leverage OOP as a transferable logic for managing healthcare socio-technical SoS complexity. By abstracting core governance roles, encapsulating local operational autonomy, modularizing system functions, and enforcing standardized interaction protocols, this approach provides a structured framework that can be analogously applied to the design and regulation of healthcare SoS, where diverse constituent systems must cooperate without relinquishing independence. To validate the proposed analogy, we implemented a Petri Net simulation in which constituent systems (e.g., hospitals, insurers, laboratories) interact via transitions governed by what we call the Confluence Interoperability Covenant (CIC). This perspective contributes to developing a scalable, interoperable governance model that is both adaptable and robust in the face of real-world complexity. The progression of the conceptual structure is summarized in Table 1 below.

Table 1. Progression of the Conceptual Structure of OOP.

Study Reference	Conceptual Structure
[34]	Introduced the concept of objects and inheritance for simulation modeling, formalizing class-based hierarchies that combine data and behavior. Established encapsulation as a means to manage complexity and reuse through hierarchical abstraction.
[36]	Pioneered the use of visual, reusable components and constraint-based modeling. Its master–instance relationships and topological constraints represented early forms of self-contained, interacting entities.
[37]	Presented a pure object-oriented paradigm where everything is an object and computation occurs through message passing. Introduced late binding, reflective metaclasses, and the idea of recursive object ontologies.
[40]	Extended OOP to procedural and industrial contexts, embedding encapsulation, inheritance, and polymorphism within scalable software environments. These languages emphasized practical modularity, reusability, and integration.
[41]	Developed a formal object calculus that defined the semantics of objects, inheritance, and polymorphism. Provided a rigorous theoretical foundation for object interaction and behavioral abstraction.
[44]	Interpreted OOP as both a technical and philosophical construct, highlighting objects as linguistic and computational abstractions that emerge through software practice. Positioned OOP as a conceptual framework for negotiating complexity and representation.
[48]	Introduced a multidimensional cognitive and structural model of OOP, emphasizing learning, abstraction, and class–object relationships as foundational to conceptual reasoning and software design.
[47]	Provided a formal comparison between nominal and structural typing via the NOOP model, clarifying how inheritance and identity shape object semantics and subtyping.
[49]	Reaffirmed the enduring relevance of OOP in AI and Big Data contexts, showing that encapsulation and polymorphism remain essential for modularity and scalability in modern computational systems.
This Study	Reinterprets OOP as a transferable conceptual logic emphasizing encapsulation, modularity, and interaction protocols. Through the CIC, the framework operationalizes these principles in a healthcare context, providing a scalable and adaptive model for SoS governance validated through simulation.

3. From Classes to Covenants: A Structural Analogy for Socio-Technical SoS Governance

With the dual insights into healthcare delivery as a socio-technical SoS and the evolving conceptual structure of OOP now established, we are positioned to explore how these domains intersect. In what follows, we formalize this analogy through a systematic mapping of OOP constructs to SoS governance mechanisms, supported by theoretical reasoning and illustrative examples.

3.1. Analogy Overview

At the heart of OOP is the concept of the interface, which defines a contract of method signatures (a specification of what methods must exist) without dictating how they should be implemented [51]. A compiler then ensures that any class claiming to implement an interface adheres to its contract, verifying that the source code conforms to the language's syntactic, semantic, and type rules. This approach closely parallels our proposition for transforming the current Collaborative healthcare SoS: introducing an external governing entity that oversees and enforces interaction protocols (i.e., shared rules and expectations)

without prescribing the internal operations of each system. In this analogy, the compiler acts as the governing authority, the interface represents a shared compliance contract, and the classes correspond to autonomous systems that implement these contracts according to their local context. This conceptual alignment reveals how OOP principles can serve as a robust design language for modeling and reasoning about governance in complex socio-technical systems.

3.2. Core Construct Mapping

In this subsection, we introduce a structured mapping between key OOP structural components and their analogues in socio-technical SoS governance. Table 2 outlines how fundamental OOP constructs as defined in [51,52]—such as classes, objects, methods, and interfaces—can be conceptually aligned with core elements of SoS architecture and governance mechanisms. Each row includes the OOP definition, its corresponding role in SoS governance, a description contextualized to complex systems, and a practical example drawn from the healthcare domain to illustrate the analogy in action.

Table 2. Mapping OOP Structural Components to SoS Governance Analogues with Healthcare Examples.

OOP Structural Component	Definition in OOP	SoS Governance Analogue	Description in the Context of SoS	Practical Healthcare Example
Class	A blueprint for creating objects that defines properties (attributes) and behaviors (methods).	Constituent System Type	Represents a type of actor or service (e.g., hospital, lab, insurer) that defines the general capabilities and role of a system.	Defines entities with care delivery, billing, and referral capabilities. For example, Hospital, Lab, Insurance, Pharmacy, etc. classes.
Object	An instance of a class with specific values assigned to attributes.	Operational System Instance	A specific real-world system operating under a constituent type.	• "Mercy General Hospital" is an instance of the Hospital class. • LabCorp Central Lab (Laboratory class) • Blue Shield of CA (Insurer class) • Walgreens #1234 (Pharmacy class)
Attribute (Field)	A variable defined in a class that holds state information.	System Configuration or Resource State	Internal parameters, data structures, or operational resources specific to a constituent system.	A hospital's bed availability, medical specialties, or patient queue status.
Method	A function associated with a class that defines its behavior.	Functional Capability	A functional service or behavior the system can perform and expose externally.	SubmitClaim(), RequestAuthorization(), or DispatchAmbulance() as functions exposed through CIC.
Interface	A contract that defines a set of method signatures without implementation.	Confluence Interoperability Covenant (CIC)	A formal governance agreement specifying required interaction points and rules for participation.	A hospital must respond to CIC messages such as • Verify Eligibility: check patient coverage via FHIR Eligibility Response. • Provide Discharge Summary: send CCD to primary care using HL7 CDA. • Report Lab Results: deliver LOINC-coded results via REST API. • Exchange Medication List: share NCPDP SCRIPT messages.
Compiler	Tool that checks whether the program conforms to language rules and type system.	External Governing Entity	Ensure all constituent systems comply with CIC terms before and during operation, without interfering in implementation.	A healthcare regulator audits interfaces and data exchange formats but does not dictate how hospitals manage internal systems. • State Health Dept.: audits FHIR payloads for schema compliance. • ONC: certifies EHR vendors against C-CDA profiles. • CMS: enforces API standards for Medicare Advantage plans. • Certifies new EHR systems through automated FHIR API validation tests.

The table above presents a structured mapping between core OOP constructs and their analogues within socio-technical SoS governance. By leveraging the structural logic of OOP, we illustrate how familiar programming components can be conceptually aligned with real-world elements in complex, decentralized systems like healthcare. Specifically,

the OOP concept of a class, which serves as a blueprint defining shared properties and behaviors, maps to a constituent system type in SoS governance. For instance, the "Hospital" class encapsulates all systems that share core capabilities such as care delivery. Its instantiations, or objects, represent specific operational entities like Mercy General Hospital or LabCorp Central Lab—each operating within the constraints and functions defined by its system type.

3.3. OOP-Inspired Principles for Socio-Technical System Governance

SoS engineering is grounded in a set of core principles, yet it continues to evolve as complexity and integration demands grow. Azani (2008) identifies several key principles essential for effective SoS design, including modularity, which enables flexible integration and replacement of components, and connectivity, which ensures communication among systems and with the environment [53]. He also highlights emergent behavior, where system-level outcomes arise from interactions among independent systems, and autonomy, which allows constituent systems to operate independently. Additionally, interdependencies and loose coupling are emphasized as crucial for maintaining stability while enabling scalability and innovation.

To advance this discourse and inform the governance and evolution of complex socio-technical SoS architectures—extending beyond structural analogies to inspire validation design—the principles of OOP can also inform the development of foundational principles for socio-technical SoS. Concepts such as abstraction, encapsulation, inheritance, polymorphism, and modularity offer valuable insights for designing interoperable, adaptable, and resilient systems.

To that end, Table 3 outlines how fundamental OOP principles can inform and advance SoS governance. Each row presents an OOP principle alongside its definition, a corresponding governance analogue, a description in the SoS context, and a practical example from the healthcare domain.

The explanation in Table 3 illustrates how OOP principles can guide the design of interoperable, flexible, and scalable governance architectures, particularly in domains like healthcare where systems are both autonomous and interdependent. In the SoS context, abstraction supports role specification without revealing internal processes, while encapsulation ensures that systems maintain operational autonomy behind standardized interfaces. Inheritance reflects the propagation of governance policies across hierarchical levels, and polymorphism captures the diverse, context-sensitive implementations of shared obligations. Modularity enables constituent systems to evolve independently while maintaining system-wide cohesion through stable interfaces.

While the OOP constructs are repurposed here as analogical building blocks for socio-technical governance, their core semantic logic is preserved through a conceptual ontology that aligns software design semantics with institutional governance structures. In this ontology, each OOP principle retains its relational meaning: encapsulation corresponds to operational autonomy with controlled access; inheritance expresses hierarchical policy propagation; and polymorphism reflects context-dependent role fulfillment under shared coordination protocols. The translation is thus structural and behavioral, not syntactic—maintaining the integrity of OOP's logic of modularity, abstraction, and bounded interaction, while adapting its semantics to the governance of autonomous socio-technical actors. This divergence underscores the critical difference between the software domain, where constructs operate within closed technical systems, and the socio-technical domain, where analogous constructs must accommodate institutional autonomy, human agency, and heterogeneous governance logics.

Table 3. OOP Principles and Their Analogues in SoS Governance with Healthcare Examples.

OOP Principle	OOP Definition	SoS Governance Analogue	SoS Description	Practical Healthcare Example
Abstraction	Exposing only relevant information and hiding unnecessary implementation details.	Governance Role Specification	Defines *what* a system is expected to do (e.g., submit insurance claims) without dictating *how* it performs the task internally.	A hospital must provide billing details via CIC, but how it retrieves and formats that data (from EHR, internal DB, etc.) is not exposed.
Encapsulation	Bundling data and methods within a class and restricting direct access to internal details.	Operational Autonomy with Controlled Interfaces	Each constituent system manages its internal workflows, data, and policies but exposes a standardized interface (CIC) for coordination.	A lab conducts diagnostics using proprietary methods but shares only validated test results and availability via a CIC interface to a hospital system.
Inheritance	A class can inherit fields and methods from a parent class, allowing reuse and extension.	Hierarchical Governance or Policy Propagation	Governance principles or service templates can be inherited from higher-level frameworks or coordinating bodies.	A regional healthcare authority mandates a common patient referral policy that individual clinics adopt and adapt to their operational context.
Polymorphism	Different classes implement the same interface in context-specific ways; behavior varies based on the actual class.	Context-Aware Role Fulfillment	Different systems fulfill the same governance role or CIC obligation using diverse methods suited to their capabilities and constraints.	Both private and public hospitals respond to an "authorized procedure" CIC request, but one routes it through a central billing system, the other through manual review.
Modularity	Structuring software into independent components (classes/modules) that can be developed and modified separately.	Constituent System Independence with Interface Cohesion	Systems can evolve independently as long as their interfaces (CICs) remain consistent; encourages plug-and-play adaptability.	An insurer updates its internal fraud detection algorithm without disrupting claim processing from providers, as the CIC protocol remains unchanged.

Having these well-defined principles establishes a coherent foundation for addressing the inherent complexity, dynamism, and heterogeneity of constituent systems. They articulate enduring guidelines that inform design decisions, governance structures, and integration strategies, thereby reducing ad hoc solutions and enabling systematic reasoning about trade-offs. Additionally, they serve as shared reference points for stakeholders—engineers, policymakers, operators—to align on objectives such as interoperability, resilience, and scalability, even as individual systems evolve independently.

3.4. The Formulation of Confluence Interoperability Covenant (CIC) as a Socio-Technical Interface

To complete the governance foundation of our proposal and leverage foundational principles, we introduce and formalize the Confluence Interoperability Covenant (CIC) as the socio-technical analogue of an object-oriented interface. We define the CIC as an adaptive, socio-technical governance artifact that codifies an integrated suite of legal, procedural, and technical standards—ranging from data-exchange formats and privacy safeguards to consent workflows and audit protocols—that define the permissible surface of interaction among constituent systems within a System-of-Systems without imposing constraints on their internal architectures or operational processes. The CIC shall be iteratively evolved through versioned amendments and shall be stewarded by a coalition of policymakers, systems engineers, legal experts, and domain practitioners to balance enforceable normative coherence with the autonomy essential for continuous innovation.

Both OOP interfaces and CICs externalize "what" interactions look like while internalizing "how" they are achieved. By delineating interaction constraints while preserving internal independence, the CIC enables diverse systems to interoperate without sacrificing innovation or contextual responsiveness. Critically, unlike code, Covenants inhabit

a broader socio-legal ecosystem: they evolve iteratively through policy cycles, involve policymakers, engineers, and legal teams, and must accommodate adaptive regulations rather than rigid syntax.

Table 4 illustrates how CIC functions as a socio-technical interface by defining standardized contracts, ensuring enforcement through external governance, and enabling interoperability without limiting internal autonomy. Just as OOP interface supports modularity and polymorphism in software, CIC allows diverse healthcare systems to integrate seamlessly, evolve independently, and fulfill shared obligations through different internal implementations. The examples demonstrate how this analogy applies to real-world scenarios, reinforcing the value of CIC in achieving structured, adaptable, and scalable governance in complex socio-technical systems.

The CIC was formulated through a detailed review of existing interoperability standards and governance protocols at the national level, encompassing frameworks such as HL7 FHIR, TEFCA, HIPAA, and ONC interoperability specifications. This synthesis ensured that the Covenant's design principles and compliance clauses are consistent with established data protection, exchange, and auditability requirements. The comprehensive list of standards, contextual interpretations, and references is provided in the Supplementary Materials accompanying this paper. In practice, the creation, validation, and versioning of the Covenant would be overseen by an external governing entity such as the U.S. HHS with its specialized agencies and offices. These entities would manage both compliance enforcement and iterative updates to the Covenant as policies, technologies, or interoperability priorities evolve. Within the simulation, this evolutionary process is represented as an adaptive function that modifies CIC parameters over time, thus reflecting the capacity for real-world governance systems to respond dynamically to change.

The CIC is designed to function within regulated, multi-actor socio-technical environments, where legal, institutional, and technical dimensions of governance intersect. In this context, under the aegis of the HHS's agencies and offices of as the external governing entity that is explained in details above in [32,33], or other existing regulatory instruments such as the General Data Protection Regulation (GDPR) [56] and the European Health Data Space (EHDS) [57] provide the macro-level principles of lawful processing, data protection, and cross-border interoperability, while the CIC operationalizes these requirements at the system interaction level. By embedding compliance clauses such as consent validation, purpose limitation, and access authorization within its interaction logic, the CIC enables compliance-by-design, ensuring that every data exchange or coordination event conforms to the prevailing regulatory and institutional context. This structure allows hospitals, insurers, laboratories, and technology vendors to maintain internal autonomy while demonstrating verifiable compliance with external mandates. This dual emphasis ensures that the healthcare SoS operates as more than the sum of its parts, achieving systemic resilience without sacrificing the agility of its components.

OOP interfaces and CICs share the conceptual goal of standardizing interactions, but they operate in fundamentally different domains are presented in Table 5. OOP interfaces are strictly technical, enforcing fixed method signatures and syntax, primarily serving developers within a controlled code environment. In contrast, CICs extend across socio-technical systems, integrating legal, procedural, and technical dimensions to coordinate diverse stakeholders, including policymakers, engineers, and legal teams. In this context, flexibility is a key differentiator: while OOP interfaces evolve through versioned APIs, CICs adapt iteratively, accommodating changing regulations, organizational policies, and emerging operational requirements. This socio-technical adaptability allows CICs to manage complexity in dynamic environments where human behavior, institutional constraints, and technology co-evolve, highlighting that effective governance in SoS depends not just

on technical standards but also on adaptive, interface-driven coordination mechanisms that bridge multiple domains and stakeholder perspectives.

Table 4. Comparative Mapping of OOP Interfaces and CIC in SoS Governance with Examples.

Aspect	OOP Interface	CIC	Examples
Purpose	Defines a contract of method signatures (what methods must exist) without implementation.	Defines contracts, protocols, and conventions (legal/technical/procedural) that systems (e.g., Providers, Insurers) must follow. Ensures heterogeneous systems (e.g., EHR platforms, billing systems) interoperate toward shared goals.	• A hospital system can update its internal EHR software as long as it adheres to the Covenant's data-sharing standards (e.g., HL7 FHIR) • A medical device supplier may use a proprietary control system internally but must expose output through DICOM or IEEE 11073 standards. (see Supplementary Materials, ref. [54]). • An insurance provider must generate claims using ANSI X12, even if it uses a homegrown system internally to manage member data.
Contract Enforcement	Enforced by the compiler/runtime: a class must implement all declared methods.	Enforced by the External Governing Entity: a system must abide by all covenant rules to remain part of the SoS.	• A regional health network may be barred from participating in national health information exchanges if it fails to support FHIR-based access tokens and audit trails • A lab's test result portal is audited by the EGE for compliance with CLIA and LOINC coding • An insurer is required to adopt CMS-mandated APIs to maintain eligibility for Medicare Advantage participation under the SoS Covenant.
Preserves Autonomy	Classes decide how to implement each method.	Systems decide how to meet each legal/procedural/technical requirement internally.	• A hospital may deploy a custom FHIR server or use a vendor-neutral archive to expose patient data, as long as it conforms to FHIR R4 profiles • A diagnostic lab may run a proprietary LIS, provided it transforms output into standardized LOINC codes for data exchange • An insurer may build a proprietary claims-transformation layer to produce ANSI X12 output as long as it adheres to the Covenant's standards.
Enables Interoperability	Any class implementing the interface can be used interchangeably by client code.	Any system adhering to the covenant can interoperate seamlessly with other governed systems.	• A telehealth provider can send encounter summaries to any EHR system that consumes FHIR CarePlan and Encounter resources • A medication supplier's inventory system communicates with diverse hospital platforms using GS1 and HL7 SPL identifiers • A pharmacy benefit manager can route prescriptions to any networked pharmacy system using the NCPDP SCRIPT standard.
Supports Modularity	New implementations can "plug in" without changing existing code.	New systems can join the SoS simply by conforming to the covenant—no redesign of the entire system is required.	• A new mobile app for chronic disease management instantly integrates with multiple provider networks by using SMART on FHIR and OAuth2 standards. (see Supplementary Materials, ref. [55]). • A startup lab can begin sending test results to existing hospital systems via HL7 v2 messages • An insurer offering value-based care enters the SoS ecosystem by mapping its proprietary KPIs into a shared quality-measure standard like HEDIS or QPP.
Promotes Polymorphism	Client code calls interface methods without knowing the concrete class.	Governance processes invoke covenant rules without needing to know each system's internal design.	• A central regulatory platform can pull patient readmission rates from all hospitals using a standardized FHIR query, regardless of whether the underlying system is Epic, Cerner, or Meditech • Emergency care data can be queried from all EMS providers via the NEMSIS API • Insurance authorization engines send pre-auth requests in the same XML format to all provider systems, regardless of how those systems validate and respond.

Table 5. Key Differences Between OOP Interfaces and CIC.

Aspect	OOP Interfaces	CIC
Scope	Technical (code)	Socio-technical (legal + procedural + technical)
Flexibility	Strict syntax (e.g., method signatures)	Adaptive (e.g., evolving regulations)
Stakeholders	Developers	Policymakers, engineers, legal teams
Evolution	Versioned APIs	Iterative governance frameworks, updated protocols, new standards, etc.

4. Functional Proof-of-Concept Simulation

To validate the proposed analogy, we developed a **functional proof-of-concept** computational simulation. It integrates systems, data standards, compliance interfaces, governance mechanisms, and simulation logic to evaluate system interoperability and policy effectiveness. This approach allows us to formally represent the constituent systems, their interactions, and the rules of the CIC to observe the emergent behavior of the entire SoS under the proposed adaptive governance regime. By simulating system interactions, policy evolution, and compliance mechanisms, it provides empirical insights into the effectiveness of governance in improving system performance, reducing fragmentation, and ensuring adherence to shared standards.

This computational simulation approach is justified by the challenges inherent in empirically examining interdependencies across large, multi-actor healthcare ecosystems. As Yin (2018) observes, large-scale case studies often face constraints related to data accessibility, institutional coordination, and analytical scope, particularly when investigating systems that span multiple jurisdictions [58]. Similarly, Flyvbjerg (2011) emphasizes that while case studies are indispensable for revealing contextual depth, their reliance on high-quality and accessible data limits their scalability in complex socio-technical domains [59]. To address these challenges, we conducted a comprehensive computational simulation representing all major healthcare subsystems, their possible interactions, and the interfaces defined by existing standards and protocols. This simulation enabled a controlled exploration of governance effects across configurations that would be impractical to observe empirically at scale. The model thus serves as an analytical proxy for large-scale, policy-driven healthcare interactions, enabling controlled experimentation that would be infeasible to observe directly in the real system.

In Einer Elhauge's seminal analysis in The Fragmentation of U.S. Health Care (2010), which attributes fragmentation to entrenched political, legal, and institutional arrangements that structurally discourage integration [60]. The governance and policy barriers produce siloed systems and proprietary data infrastructures, where hospitals, payers, and laboratories maintain separate records and operational protocols, leading to inefficiencies, duplicated efforts, and inconsistent outcomes. In this context, fragmentation represents the absence of unified governance—a proliferation of autonomous, self-regulated entities whose interactions remain episodic and weakly coordinated. Within the simulation, this condition is captured through quantifying the degree of disconnection among constituent systems. A high fragmentation score thus corresponds to the real-world phenomenon described by Elhauge: a healthcare ecosystem characterized by policy-driven isolation, institutional autonomy, and the lack of systemic cohesion.

4.1. Simulation Architecture and Key Components

The simulation framework is implemented in Python 3.7 and utilizes several key classes and functions to represent the SoS dynamics. It uses a Petri-Net style representation of an interacting SoS to model the structural and behavioral consequences of governance at the interaction layer. Each node represents a class of autonomous actors (e.g., hospital, insurer, laboratory, pharmacy, ICT vendor). Nodes have internal state variables (e.g., compliance propensity, backlog, local policy) and expose an interface through which interactions are negotiated. Transitions represent discrete interaction events (data exchange, authorization request, billing messages). The CIC is modeled as a set of conditional rules that are evaluated at each attempted transition; these rules can block, allow, or reroute transactions and can update local state (penalties, compliance learning). The architecture integrates governance, compliance, and stochastic interaction mechanisms to capture the dynamics of system interoperability under varying levels of policy enforcement.

System Representation: Each system is instantiated as an EnhancedCICPlace, extending a base Place class to incorporate actor type (human, AI, or hybrid), compliance strategy (manual, automated, or blockchain), implemented interface methods, audit score, and quarantine status. Data exchanges are represented via Token objects, which carry information with a defined format and sensitivity level.

Governance Mechanism: A GovernanceEntity enforces compliance through a set of policies (Compliance Interface Contracts, CICs) and maintains system-specific requirements. Policies can evolve dynamically during the simulation, for example by adding consent workflows or human override requirements. Systems adopt required methods probabilistically, reflecting imperfect compliance adoption in realistic scenarios.

Interaction Dynamics: Interactions between systems are defined in a structured INTERACTIONS set, specifying source, target, and required interface methods. At each simulation step, interactions are randomly sampled, and compliance is verified for both source and target systems. Successful interactions are recorded in a dynamic interaction graph, while failures and violations are tracked to evaluate system performance.

Auditing and Fragmentation: Periodic audits assess system compliance, updating audit scores and potentially quarantining non-compliant systems. Network fragmentation is calculated as the normalized count of disconnected components in the interaction graph, providing insight into the structural coherence of the SoS.

Visualization and Analysis: The framework includes visualization tools for network topology, policy evolution, interaction flows, compliance heatmaps, audit timelines, and distributions of actor types and compliance strategies. These outputs support comparative analysis of governed versus ungoverned scenarios and facilitate interpretation of systemic behaviors under different governance and compliance conditions.

4.2. Interpretation and Analytic Value

The simulation was executed under two primary conditions: with and without governance, to facilitate a comparative analysis of the CIC's impact. The _run_tests() function confirmed the foundational integrity of the simulation setup, including system presence and full adoption under ideal governed conditions.

4.3. Comparative Simulation Outcomes

Running deterministic comparative simulations for 400 steps yielded the following key statistics as presented in Table 6:

Table 6. Comparative Metrics of System Performance with and without Governance Mechanisms.

Metric	With Governance	Without Governance
Successes	339	68
Failures	61	332
Fragmentation	0.1538	0.8077
Policy Updates	1	0
Violations	36	197

These results demonstrate a significant difference in system performance. The governed scenario exhibited a substantially higher number of successful interactions (339 vs. 68) and a drastically lower fragmentation rate (0.1538 vs. 0.8077). Fragmentation, defined as the ratio of failed to total interactions, serves as a critical indicator of interoperability challenges. The single policy update in the governed scenario indicates the adaptive nature of the CIC, while the recorded violations highlight the enforcement aspect and the systems' initial non-compliance before adaptation.

What the simulation demonstrates. The experimental design contrasts two principal conditions: (1) No governance (baseline), which models interactions without an enforceable interface contract and (2) CIC governance, where all interaction attempts are evaluated and enforced against CIC clauses. The simulation therefore tests the governance function —that is, whether the presence of an enforceable, interface-level covenant improves systemic connectivity, reduces fragmentation, and stabilizes compliance dynamics. Importantly, this experiment does not claim to represent every possible governance model. Instead, it isolates the causal effect of embedded, enforceable governance at the interaction boundary versus its absence. Conceptually, CIC acts as a proxy for the operationalization of governance (regardless of institutional embodiment) and is therefore a representative test of the value of executable contractual governance. Results should be interpreted as evidence that executable interface governance can materially improve network cohesion; they do not (in this version) demonstrate that CIC is superior to every alternative governance arrangement—a point we discuss below and propose to investigate in future work.

We would like to emphasize that the simulation contrasts two system configurations: one without governance and one under CIC-based governance. This binary comparison intentionally abstracts away from the institutional variations of governance models (such as data trusts [61] or static data contracts [62]) to isolate the governing effect itself. The goal is to evaluate whether the presence of a formalized, enforceable interaction covenant—regardless of its institutional embodiment—improves system-level coordination and reduces fragmentation. Alternative models such as data trusts or data contracts differ primarily in where and how governance is expressed (institutional or contractual), whereas this simulation tests what governance achieves when embedded at the interaction layer. Consequently, the "without vs. with CIC" comparison serves as a representative benchmark for the systemic impact of governance enforcement, independent of its specific legal or organizational form. Hence, the results can be interpreted as indicative of governance-enabled interaction performance broadly, rather than exclusive to the CIC implementation.

The simulation does not employ data-driven training or optimization but rather conducts systematic parameter calibration to evaluate network behavior under different governance configurations. The objective is to observe how the introduction of the CIC influences structural cohesion, compliance stability, and network resilience. Network-level parameters were analyzed to quantify and interpret systemic fragmentation. This design allows the simulation to reflect realistic variations in institutional autonomy and governance enforcement without relying on data re-use or tuning against an evaluation subset.

This network visualization illustrates the successful interactions among heterogeneous healthcare entities, with nodes representing individual systems (e.g., hospitals, insurers, diagnostic labs, patients) and edges indicating completed interactions that complied with the required interface methods (CICs), as illustrated in Figure 3. Node colors correspond to the compliance strategy adopted by each system: manual (green), automated (blue), and blockchain (red). The Hospital node occupies a central hub position, reflecting its role as a major connector across multiple categories, including insurance, diagnostic labs, and support services. Similarly, nodes like Pharma, Equipment_Distributor, and Medical_Transport form secondary hubs linking specialized services to clinical care providers. Notably, peripheral but essential actors—such as cleaning services, guardians, and transport—are explicitly connected through governed interactions, reinforcing the socio-technical scope of the model. The distribution of links reveals polymorphic interaction modes, such as automated device registration, blockchain-verified claims, and manual review of sensitive processes like insurance claims, highlighting how heterogeneous validation ensures both interoperability and contextual trust.

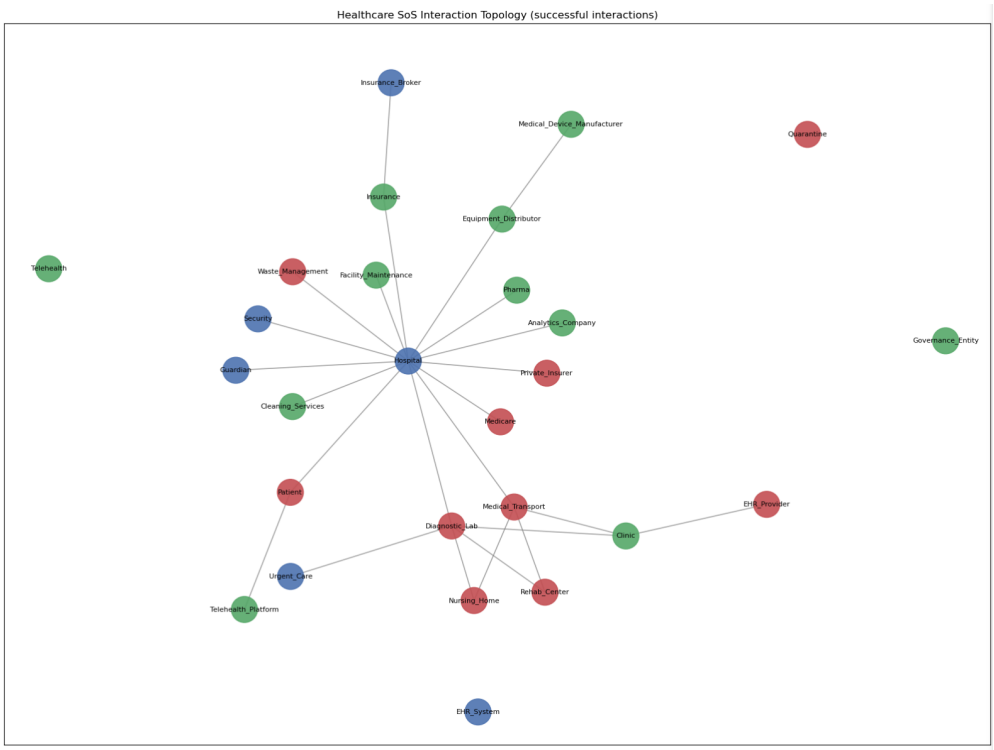


Figure 3. Healthcare SoS Interaction Topology.

Figure 4 depicts the incremental adaptation of CIC policies across iterations. The x-axis represents policy iterations, while the y-axis reflects the versioning of the CIC. The linear progression from version 1.00 to 1.10 demonstrates how governance mechanisms evolve incrementally in response to operational feedback, recorded violations, or emerging systemic needs. Technically, this progression is triggered by feedback loops embedded in the simulation: violations, audit outcomes, or performance degradations flag the need for contract revision, which then increments the CIC version. Each policy iteration incorporates lessons learned from prior enforcement and system performance, thereby enabling continuous improvement. Conceptually, this evolution underscores the dual role of CICs: they serve as externalized constraints that enforce compliance, while simultaneously functioning as adaptive governance instruments capable of continuous refinement. This shows the model successfully captures a feedback loop where the state of the SoS directly influences the evolution of its own governing covenant, enhancing the overall system's resilience and ability to manage complexity. Consequently, the figure reinforces the idea that CIC-based governance supports not only compliance assurance but also sustainable adaptability within complex healthcare SoS.

Figure 5 illustrates the temporal evolution of compliance audit scores across constituent systems within the CIC-governed healthcare System-of-Systems. The compliance audit scores illustrate natural fluctuations and gradual degradation over time, reflecting the effects of random perturbations that simulate real-world operational drift. By jointly visualizing audit scores with observed violations, the figure contextualizes system-level resilience and fragility: resilient systems sustain high scores with minimal violations, whereas fragile ones exhibit sharp declines or persistent underperformance. The time-series traces further highlight the impact of governance interventions—such as quarantining or policy adjustments—on system trajectories, distinguishing between cases of recovery, stabilization, or continued failure. These dynamics substantiate the proposition that CIC provides a governance mechanism that enables external monitoring and targeted remediation while respecting internal autonomy. Importantly, the model illustrates that governance is not

purely punitive; rather, it is structured to uphold systemic integrity by permitting graceful degradation and subsequent recovery among constituent systems. In doing so, the governance framework demonstrates its capacity to adaptively maintain overall performance while accommodating heterogeneity in operational behaviors across the healthcare system-of-systems.

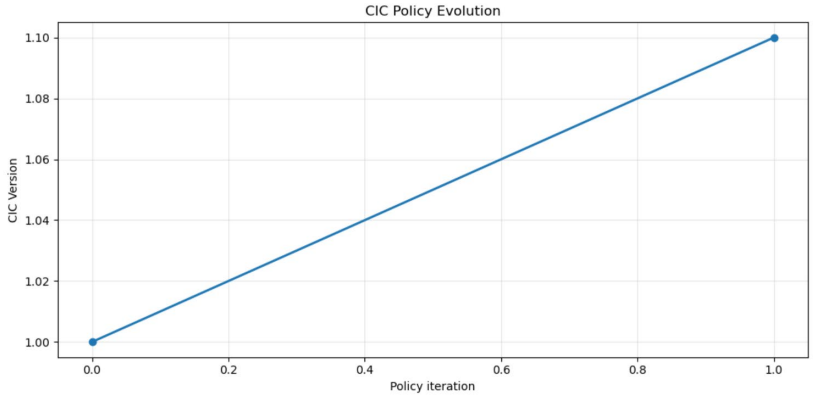


Figure 4. CIC Policy Evolution.

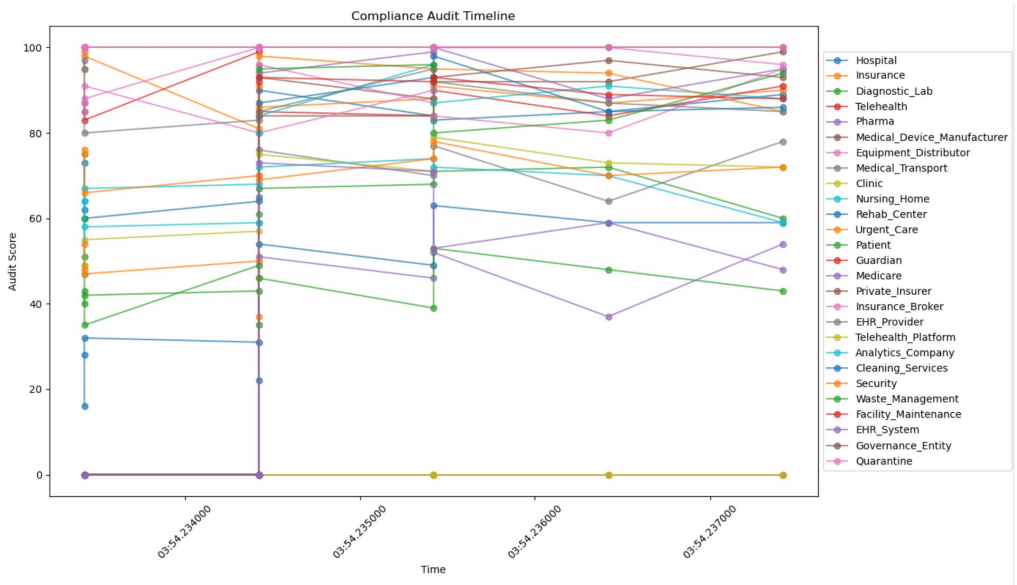


Figure 5. Compliance Audit Timeline.

Figure 6 presents the distribution of successfully executed interactions across different system-to-system processes within the simulated healthcare System-of-Systems. Each bar corresponds to a distinct interaction type, and the x-axis indicates the frequency of successful executions. In the governance context, this distribution highlights where CIC and monitoring mechanisms exert the most influence. High-frequency interactions are natural leverage points for enforcing compliance and ensuring interoperability, since failures here propagate widely. Conversely, low-frequency interactions may escape attention but represent potential hidden vulnerabilities if neglected. Thus, the figure illustrates how the simulation not only captures interaction diversity but also provides a data-driven foundation for prioritizing governance interventions based on systemic impact.

The analyses of actor types and compliance strategies confirm the heterogeneity of the SoS. The distribution reflects a realistic ecosystem. The fact that the simulation runs successfully with this mix proves that the CIC interface provides an effective abstraction layer. It allows vastly different types of actors (e.g., an AI-powered Diagnostic

Lab using automated validation and a human-operated Clinic using manual review) to interoperate seamlessly, as long as they adhere to the commonly defined interaction surface, as illustrated in Figure 7. Additionally, Frequency counts of different transitions (manual/automated/blockchain) show how often each implementation path is used. If automated paths dominate throughput while manual paths correspond to higher enforcement or longer quarantine episodes, that provides evidence for trade-offs between automation (throughput) and governance (auditability, safety). Furthermore, Distribution plots allow linking actor types (human/AI/hybrid) to observe compliance performance and choices of implementation strategy. For example, AI-dominated systems implementing AUTOMATED_VALIDATION may show faster throughput but sensitivity-related violations, motivating CIC provisions requiring human_override for sensitive data.

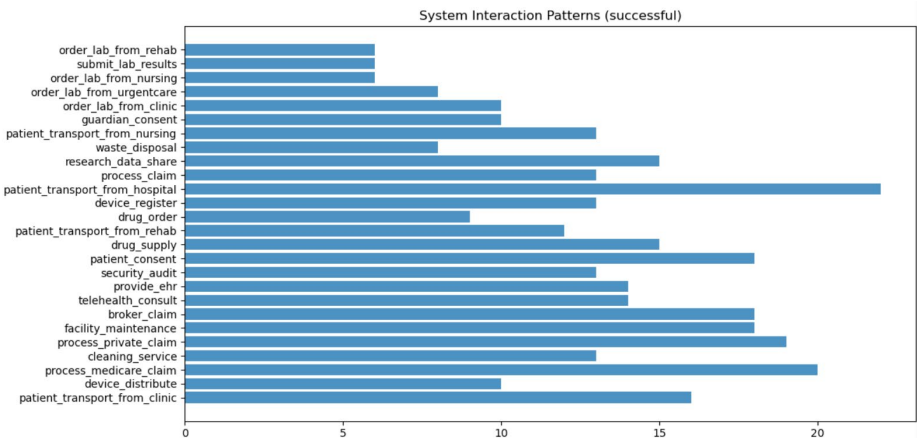


Figure 6. System Interaction Patterns.

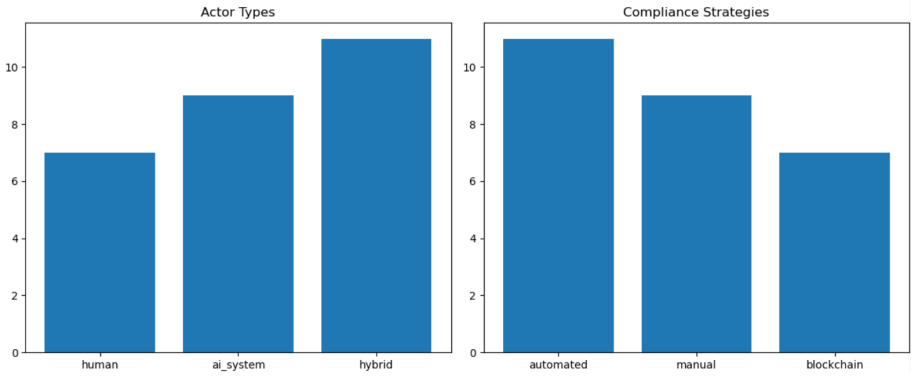


Figure 7. Actor Types and Compliance Strategies.

5. Discussion

This paper contributes to the field of Systems Engineering, especially SoS, by advancing a governance-oriented architectural paradigm for socio-technical SoS, with healthcare serving as a representative domain. By drawing a structural and functional analogy between OOP principles and SoS governance mechanisms, the study bridges conceptual clarity with practical application. The CIC, proposed as a socio-technical analogue to OOP interfaces, functions as a governance mechanism and formalizes a modular, enforceable, and adaptive coordination layer among autonomous systems.

The discussion of OOP principles—such as encapsulation, polymorphism, inheritance, and modularity—in a governance context allows for the abstraction of governance logic from implementation-specific details, thus enabling scalable system integration without compromising local autonomy. This approach reflects a shift from rigid, prescriptive control

models to flexible, contract-based governance mechanisms, offering a robust blueprint for achieving interoperability, resilience, and policy conformance across heterogeneous systems. The mapping tables and case examples provide actionable guidance for translating software engineering logic into socio-technical design patterns that are both rigorous and adaptable.

The simulation provides a concrete instantiation of the concepts and demonstrates the practical value of framing SoS governance through the lens of OOP theory. However, the true significance of this work lies not in the simulation itself, but in the powerful conceptual paradigm shift that the OOP-SoS isomorphism represents for the governance of complex, heterogeneous systems. This isomorphism provides a tool for managing this complexity through the core OOP principles of abstraction and encapsulation.

Abstraction: The CIC is an abstraction layer. A Telehealth_Platform developer needs only understand the CIC-mandated ISO/TS 13131 data format (see Supplementary Materials, ref. [63]) to initiate a consultation. They are abstracted away from the internal complexities of the Patient's device, their Insurance provider's claims adjudication logic, or the Hospital's patient record system. This dramatically reduces the cognitive and technical load required for integration.

Encapsulation: Each CIC element in our model is a black box. Its audit_score, its choice of actor_type, and its internal logic for token generation are encapsulated. The governor and other systems cannot see inside, nor do they need to. They interact solely with the public interface. This encapsulation is what makes it possible for an AI-driven Diagnostic_Lab and a human-operated Clinic to coexist and interoperate within the same SoS. Their irreducibly different internal natures are rendered irrelevant by the common interface.

From a governance perspective, this analogy is powerful because it reframes compliance from a purely regulatory burden into an architectural design mechanism. The simulation shows that this approach is not just theoretically sound but practically executable. By codifying interaction rules as explicit constraints enforced at system boundaries, SoS governance can achieve predictable interoperability and traceable accountability without centralizing or homogenizing internal architectures. For example, the governing entity's ability to dynamically update CIC versions in response to load or violations shows that interfaces can evolve adaptively, enabling a balance between stability and innovation. Additionally, the governance entity can effectively audit and enforce compliance based solely on this external interface, without any knowledge of, or control over, the encapsulated internals. This capability parallels versioned software interfaces, where backward-compatible updates maintain ecosystem coherence while allowing incremental improvements.

It is crucial to emphasize that the CIC *is more than a set of rules*; **it is a formal mechanism** for instantiating structured autonomy, a principle that has long been a theoretical goal but a practical challenge in SoS engineering. Instead of dictating internal implementation, the CIC-as-interface paradigm externalizes the contract for interaction. The guard conditions on the transitions (e.g., x.data_format == 'FHIR') do not care how a Hospital system produces an FHIR-formatted token—whether through a commercial EHR, an in-house solution, or a future quantum computing module. This is directly analogous to an OOP interface demanding a method return an Integer without specifying the algorithm used. This abstraction is the bedrock of structured autonomy: it grants constituent systems the freedom to innovate and optimize internally ("how") while providing the SoS governor with ironclad guarantees about behavior at the boundaries ("what"). In software design, an interface specifies the required signatures of methods while allowing implementers to freely choose internal logic. Thereby, CIC offers a systematic way to ensure that constituent systems can engage in standardized interaction without requiring centralization or uniformity.

The polymorphic choice of compliance strategies observed in the simulation is a direct manifestation of this autonomy in action.

The simulation results reveal that governance functions not merely as an external constraint but as fundamental infrastructure for emergent coordination in complex adaptive systems. The stark contrast between failure rate without governance and success rate with governance represents more than quantitative improvement—it demonstrates a phase transition from disorder to order, analogous to crystallization or network percolation thresholds. The policy update from CIC_v1.0 to CIC_v1.1, triggered by 36 violations that themselves become informative signals rather than pure failures, illustrates governance as a self-organizing feedback system where the covenant evolves through interaction with the very system it governs. The Governance Entity in our simulation does not merely punish deviation; it learns from the state of the SoS and adapts the covenant. This challenges traditional regulatory paradigms that treat rules as static impositions and instead positions CICs as living protocols that learn from operational reality. Moreover, the polymorphic compliance strategies—manual, automated, and blockchain—coexisting within a unified governance framework demonstrate that effective coordination does not require homogenization but rather a sufficiently abstract interface layer that preserves heterogeneity while ensuring interoperability. This has profound implications: governance architectures should be designed not to eliminate diversity but to make diversity composable, enabling AI-powered diagnostic systems, human-operated clinics, and blockchain-secured insurers to interact seamlessly precisely because the CIC defines the interaction surface without dictating implementation internals.

The most consequential insight lies in how CIC governance achieves scalable trust in the absence of centralized authority—addressing what is perhaps the fundamental challenge of digital healthcare ecosystems. The reduction in fragmentation occurs not through hierarchical control but through a shared covenant that constituent systems voluntarily adopt and enforce upon themselves, with violations serving as system-level immune responses that trigger adaptation rather than punishment. The compliance audit timelines revealing "graceful degradation and subsequent recovery" demonstrate that resilient systems are not those that never fail but those that fail gracefully, learn quickly, and recover reliably. This has transformative implications for healthcare interoperability policies, which have historically failed by attempting to mandate technological uniformity (failed interoperability standards) or by leaving coordination entirely to market forces (persistent fragmentation). The CIC model suggests a third path: governance at the interaction layer that provides just enough structure to enable coordination while preserving sufficient autonomy to enable innovation. As healthcare systems increasingly incorporate AI, IoT devices, blockchain registries, and human judgment into hybrid workflows, the ability to govern heterogeneous interactions without requiring technological consensus becomes not merely advantageous but existentially necessary. The simulation's success with realistic ecosystem diversity proves that such governance can scale across the full socio-technical spectrum, suggesting that CIC-based approaches could finally break the interoperability deadlock that has plagued healthcare for decades by shifting focus from "how systems work internally" to "how systems interact externally".

The escalation of policy in response to violations models a learning feedback loop. This is a form of meta-governance: governing the rules of governance themselves. By updating the interface, the governor can steer the entire SoS toward more desirable emergent properties without ever directly commanding a single system to change its code. Systems are forced to adapt their internal implementations to comply with the new interface version to remain part of the SoS, thus achieving system-wide change through a decentralized, evolutionary pressure rather than a centralized mandate. This creates a resilient, anti-fragile

governance model. The SoS becomes capable of not just withstanding shocks but of learning from them and evolving to be better, a critical capability for socio-technical systems operating in dynamic environments.

The proposed framework demonstrates inherent scalability by ensuring that the addition of new constituent systems does not proportionally increase the complexity of the overall SoS. In conventional architectures, growth in the number of independent systems and their interrelationships often leads to exponential integration challenges and coordination overhead. By contrast, the use of the CIC establishes a standardized governance and interaction layer that decouples system autonomy from system integration. This design allows new systems to be seamlessly "plugged in" as long as they conform to the CIC, thereby preserving interoperability without requiring bilateral adaptations with every other participant. Equally important, systems that fail to comply with the covenant can be excluded, preventing them from disrupting collective performance or contributing to fragmentation. Beyond technical scalability, the framework also advances digital integration by harmonizing heterogeneous platforms, data standards, and workflows into a coherent ecosystem. This integration not only enhances efficiency but also enables collective social improvement, as interoperable healthcare systems can more effectively share knowledge, coordinate care, and respond to public health needs. In this way, the framework supports sustainable expansion of the SoS while simultaneously fostering societal resilience, equity, and improved health outcomes.

The implications of this framework extend across multiple dimensions of healthcare delivery, offering healthcare organizations a pathway to interoperability that preserves institutional autonomy while enabling meaningful data exchange and care coordination through interface standardization rather than internal process harmonization. From a regulatory perspective, the external governing entity model provides a mechanism for ensuring compliance without micromanaging operational details, aligning with contemporary trends toward outcome-based regulation and performance-based accountability, while the iterative evolution of CIC through versioned amendments reflects the dynamic nature of healthcare regulation and technology. The framework's modularity principle has significant implications for healthcare innovation by enabling new systems to integrate through covenant compliance rather than wholesale system redesign, lowering barriers to entry for innovative technologies and service models, and accelerating the adoption of emerging technologies such as artificial intelligence, telemedicine, and precision medicine tools within existing healthcare ecosystems.

From a theoretical standpoint, the findings extend SoS governance discourse by linking formal governance constructs to measurable outcomes, showing that structural design, actor compliance, and interaction policies collectively shape system-level performance. The inclusion of socio-technical heterogeneity and polymorphic interactions emphasizes that governance must account for diverse capabilities, objectives, and compliance behaviors, as these variations directly affect the propagation of coordination and fragmentation across the network. Practically, the approach provides a versatile tool for stakeholders in healthcare and other complex infrastructures to evaluate, test, and refine governance strategies prior to implementation, reducing trial-and-error risk while enhancing adaptability to evolving system requirements. Overall, the study underscores the critical importance of embedding adaptive, interface-driven governance mechanisms to mitigate fragmentation, optimize interdependencies, and ensure robust, resilient performance in multi-actor, multi-layered systems.

Although this research focuses on healthcare as a representative case of Collaborative SoS, the framework is grounded in the shared structural characteristics that define SoS configurations more broadly. Across domains (e.g., transportation, energy, defense, or

smart cities) the common theme lies in the interdependencies and complementarities that allow autonomous systems to coordinate and collectively achieve outcomes unattainable by any single system. As discussed in the related work section, these properties form the universal foundation of SoS theory. The primary differences among SoS types emerge from the nature of these interdependencies and the degree of constituent autonomy, which vary by domain and governance context. Accordingly, while the present study applies the CIC within the healthcare sector, the structural insights and governance mechanisms it reveals are transferable to other industries. However, realizing such transfer requires systemic and contextual analysis of each sector's interaction modalities, institutional arrangements, and autonomy boundaries.

While the OOP analogy provides valuable insights, several methodological considerations warrant attention, as the translation of programming concepts to socio-technical systems involves inherent limitations due to fundamental differences between computational and social contexts, where healthcare organizations operate within complex social, economic, and political environments that introduce variability and unpredictability not present in programmatic contexts. The analogy between OOP and healthcare SoS serves as a structural abstraction that formalizes coordination logic without diminishing the human, institutional, and political dimensions of healthcare. While the OOP framework provides transferable design insights, its application in this study is limited to structuring socio-technical interactions. The enduring realities of professional autonomy, organizational hierarchy, and policy-driven incentives remain essential forces that shape governance outcomes, as discussed in this section. The enforcement mechanisms available to external governing entities differ significantly from compiler-based validation in software systems, as governance entities must navigate ambiguous regulatory interpretations, resource constraints, and political considerations rather than definitively determining interface compliance, suggesting that the CIC framework may require more sophisticated monitoring and enforcement mechanisms than the direct analogy implies. Furthermore, the framework's emphasis on standardized interfaces may inadvertently create new forms of rigidity, where healthcare organizations find themselves adapting their internal processes to accommodate interface requirements, potentially limiting the very autonomy the framework seeks to preserve despite its intention to enable innovation through interface standardization.

Real-world implementation would face socio-technical challenges that extend beyond architecture design. The cost of integration—including the alignment of legacy IT systems, retraining of personnel, and upgrading of compliance infrastructures—represents a substantial investment for most healthcare organizations. Moreover, resistance from healthcare workers may emerge from perceived threats to professional autonomy or increased administrative burden, particularly when new interoperability rules affect established clinical workflows. These challenges are not unique to this framework; rather, they reflect broader issues inherent to digital transformation and governance enforcement in complex socio-technical systems. Addressing them requires phased implementation strategies, strong institutional incentives, and stakeholder co-design processes that ensure governance mechanisms like the CIC enhance rather than constrain clinical practice. Future research will explore these adoption pathways, cost–benefit analyses, and the development of participatory governance models to facilitate sustainable implementation.

Although this study primarily emphasizes the architectural and computational aspects of governance in healthcare SoS, the views of stakeholders such as clinicians, patients, administrators, policymakers, etc. are essential for real-world adoption. Governance cannot be sustained purely through technical or institutional design; it must also account for the social, ethical, and behavioral realities that shape how systems and people interact. For example, clinicians may prioritize workflow efficiency and autonomy, patients may be

more concerned with privacy and consent, and administrators often focus on cost and regulatory compliance. Each of these perspectives influences the acceptability and legitimacy of governance mechanisms. Incorporating these perspectives through participatory design, stakeholder consultation, and user-centered evaluation will be crucial for ensuring that governance mechanisms enhance collaboration and trust rather than imposing additional burdens.

6. Conclusions

This study has advanced the discourse on healthcare delivery governance by conceptualizing the system as a socio-technical SoS and drawing a structural analogy to the principles of OOP. Through this lens, we demonstrated that constructs such as encapsulation, polymorphism, and modularity provide a transferable logic for governing autonomous yet interdependent entities, offering both theoretical rigor and practical applicability. The introduction of the CIC formalizes this analogy, serving as an interface-like governance artifact that defines enforceable rules for interaction while safeguarding local autonomy. Complemented by Petri Net simulations, our framework illustrates how an external governing entity can act as a compiler—enforcing compliance, promoting interoperability, and steering emergent dynamics toward alignment with overarching health objectives. These findings advance SoS governance theory by linking formal governance constructs to measurable effects and provide a practical framework for designing adaptive, evidence-based policies. Beyond healthcare, this work offers a blueprint for managing complex infrastructures where autonomous yet interdependent actors must align toward shared goals, emphasizing the value of interface-driven, adaptive governance in mitigating fragmentation and enhancing systemic robustness. The CIC provides a conceptual framework that balances autonomy with alignment, enabling emergent coordination without sacrificing operational independence through its integrated legal, procedural, and technical dimensions.

While this research lays critical conceptual and methodological foundations, several avenues remain open for further exploration. First, empirical validation is necessary to test the CIC's feasibility across real-world healthcare ecosystems, particularly where heterogeneous data standards, institutional incentives, and policy mandates intersect. Empirical validation will require the development of a formal CIC specification that integrates all necessary healthcare standards and protocols—including HL7 FHIR, DICOM, IHE profiles, HIPAA compliance frameworks, and relevant quality measures—followed by pilot implementations in real-world healthcare networks to assess practical effectiveness, transition costs, and organizational impacts. Second, focus on examining the various types of interdependencies among independent systems within healthcare to better understand their structural and behavioral characteristics, and to identify the specific coordination and compliance mechanisms required for their effective inclusion within the CIC framework. Third, future work could extend our graph-theoretic and Petri Net analyses into agent-based models or digital twin architectures, enabling richer simulations of adaptation, negotiation, and resilience under dynamic governance rules. Fourth, future work could also benefit from ontology-based modeling using the Web Ontology Language (OWL) combined with graph knowledge representations, enabling the formal capture of interdependencies, semantic relationships, and governance rules among healthcare systems. Such a digital representation would facilitate automated reasoning, seamless integration across platforms, and dynamic policy evolution, allowing CIC and governance mechanisms to be continuously updated and digitally aligned with emerging regulations, system behaviors, and healthcare delivery needs. Finally, incorporating machine learning–driven compliance monitoring

and blockchain-based covenant enforcement could enhance the technical robustness of the CIC, providing mechanisms for automated auditing and trust assurance at scale.

Supplementary Materials: The supporting information can be downloaded at: https://www.mdpi.com/article/10.3390/systems13121113/s1. The supplementary materials include detailed establishment and derivation of the CIC; and the complete simulation code workflow and implementation details. References [54,55,63–78] are cited in the Supplementary Materials.

Author Contributions: Conceptualization, M.M. (Mohamed Mogahed) and M.M. (Mo Mansouri); methodology, M.M. (Mohamed Mogahed) and M.M. (Mo Mansouri); software, M.M. (Mohamed Mogahed); validation, M.M. (Mohamed Mogahed) and M.M. (Mo Mansouri); formal analysis, M.M. (Mohamed Mogahed) and M.M. (Mo Mansouri); resources, M.M. (Mo Mansouri); data curation, M.M. (Mohamed Mogahed); writing—original draft preparation, M.M. (Mohamed Mogahed); writing—review and editing, M.M. (Mohamed Mogahed) and M.M. (Mo Mansouri); visualization, M.M. (Mohamed Mogahed); supervision, M.M. (Mo Mansouri); project administration, M.M. (Mo Mansouri). All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: This study is based on computational simulations and theoretical framework development. No empirical data were collected. The simulation code workflow is provided in the Supplementary Materials.

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

References

1. Marjanovic, S.; Altenhofer, M.; Hocking, L.; Chataway, J.; Ling, T. Innovating for improved healthcare: Sociotechnical and innovation systems perspectives and lessons from the NHS. *Sci. Public Policy* **2020**, *47*, 283–297. [CrossRef]
2. Kwiatkowska, M. Recommender Systems in Healthcare: A Socio-Technical Systems Approach. In *Recommender Systems for Medicine and Music*; Ras, Z.W., Wieczorkowska, A., Tsumoto, S., Eds.; Studies in Computational Intelligence; Springer: Cham, Switzerland, 2021; Volume 946, pp. 15–30. [CrossRef]
3. Buntin, M.B. Confronting Challenges in the US Health Care System: Potential Opportunity in a Time of Crisis. *JAMA* **2021**, *325*, 1399–1400. [CrossRef] [PubMed]
4. Benson, T.; Grieve, G. *Principles of Health Interoperability: FHIR, HL7 and SNOMED CT*, 4th ed.; Health Information Technology Standards; Springer: Cham, Switzerland, 2020. [CrossRef]
5. Osamika, D.; Adelusi, B.S.; Kelvin Agwu, M.T.C.; Mustapha, A.Y.; Forkuo, A.Y.; Ikhalea, N. A Critical Review of Health Data Interoperability Standards: FHIR, HL7, and Beyond. *World Sci. News* **2025**, *203*, 194–233.
6. Baxter, G.; Sommerville, I. Socio-Technical Systems: From Design Methods to Systems Engineering. *Interact. Comput.* **2011**, *23*, 4–17. [CrossRef]
7. Behymer, K.J.; Flach, J.M. From autonomous systems to sociotechnical systems: Designing effective collaborations. *She Ji J. Des. Econ. Innov.* **2017**, *2*, 105–114. [CrossRef]
8. Cardoso, R.C.; Logan, B.; Meneguzzi, F.; Oren, N.; Yun, B. Resilience, Reliability, and Coordination in Autonomous Multi-Agent Systems. *AI Commun.* **2022**, *35*, 339–356. [CrossRef]
9. Sawaragi, T. Design of resilient socio-technical systems by human-system co-creation. *Artif. Life Robot.* **2020**, *25*, 219–232. [CrossRef]
10. Heo, K.; Yi, S. (De)centralization in the Governance of Blockchain Systems: Cryptocurrency Cases. *J. Organ. Des.* **2023**, *12*, 59–82. [CrossRef]
11. Carayon, P.; Bass, E.; Bellandi, T.; Gurses, A.; Hallbeck, S.; Mollo, V. Socio-Technical Systems Analysis in Health Care: A Research Agenda. *IIE Trans. Healthc. Syst. Eng.* **2011**, *1*, 145–160. [CrossRef] [PubMed] [PubMed Central]
12. Papavasiliou, S.; Gorod, A.; Reaiche, C. System of systems engineering governance framework for digital transformation: A case study of an Australian large government agency. *Syst. Eng.* **2023**, *27*, 267–283. [CrossRef]
13. Sittig, D.F.; Singh, H. A new socio-technical model for studying health information technology in complex adaptive healthcare systems. *BMJ Qual. Saf.* **2010**, *19* (Suppl. 3), i68–i74. [CrossRef]
14. Carayon, P. Sociotechnical systems approach to healthcare quality and patient safety. *Work* **2012**, *41* (Suppl. 1), 3850–3854. [CrossRef] [PubMed]

15. Dong, R.; Sahu, A.; Black, A. Governance in the transformational journey toward integrated healthcare: The case of Ontario. *J. Inf. Technol. Teach. Cases* **2024**, *14*, 24–36. [CrossRef]
16. Bennaceur, A.; Stuart, A.; Price, B.A.; Bandara, A.; Levine, M.; Clare, L.; Cohen, J.; Mccormick, C.; Mehta, V.; Bennasar, M.; et al. Socio-Technical Resilience for Community Healthcare. In TAS '23: Proceedings of the First International Symposium on Trustworthy Autonomous Systems, Edinburgh, UK, 11–12 July 2023; ACM: Edinburgh, UK, 2023; pp. 1–12. [CrossRef]
17. de Savigny, D.; Adam, T. (Eds.) *Systems Thinking for Health Systems Strengthening*; World Health Organization: Geneva, Switzerland, 2009; Available online: https://iris.who.int/handle/10665/44204 (accessed on 1 November 2025).
18. Ansari, M.M.; Kumar, S.; Bin Heyat, M.B.; Ullah, H.; Bin Hayat, M.A.; Sumbul; Parveen, S.; Ali, A.; Zhang, T. SVMVGGNet-16: A Novel Machine and Deep Learning Based Approaches for Lung Cancer Detection using Combined SVM and VGGNet-16. *Curr. Med. Imaging* **2025**, *21*, e15734056348824. [CrossRef] [PubMed]
19. Ahmed, N.; Rozina Ali, A.; Raziq, A. Images denoising for COVID-19 chest X-ray based on multi-scale parallel convolutional neural network. *Multimed. Syst.* **2023**, *29*, 3877–3890. [CrossRef]
20. Castelle, K.; Bradley, J.M.; Chesterman, C.W. Systems Theory for Complex System Governance. In *Complex System Governance*; Keating, C.B., Katina, P.F., Chesterman, C.W., Jr., Pyne, J.C., Eds.; Topics in Safety, Risk, Reliability and Quality; Springer: Cham, Switzerland, 2022; Volume 40, pp. 57–74. [CrossRef]
21. Mogahed, M.; Mansouri, M. A new horizon for healthcare delivery: A system of systems perspective and governing proposition. In Proceedings of the 2024 INCOSE International Symposium, Dublin, Ireland, 2–6 July 2024; Volume 34, pp. 1674–1686. [CrossRef]
22. Falegnami, A.; Tomassi, A.; Corbelli, G.; Romano, E. Managing Complexity in Socio-Technical Systems by Mimicking Emergent Simplicities in Nature: A Brief Communication. *Biomimetics* **2024**, *9*, 322. [CrossRef] [PubMed] [PubMed Central]
23. Mogahed, M.; Mansouri, M. Navigating the Complexities of Systems of Systems: Analysis, Challenges, and a Novel Proposed Construct. In Proceedings of the 2024 IEEE International Symposium on Systems Engineering (ISSE), Perugia, Italy, 16–18 October 2024; pp. 1–8. [CrossRef]
24. Mogahed, M.; Mansouri, M. Systems of systems governance frameworks: A thorough scoping review and synthesis towards enhancing healthcare delivery. *IEEE Syst. J.* **2025**, *19*, 648–658. [CrossRef]
25. Calida, B.Y.; Keating, C.B. System Governance: Emergence of Practical Perspectives Across the Disciplines. In *Infranomics*; Gheorghe, A., Masera, M., Katina, P., Eds.; Topics in Safety, Risk, Reliability and Quality; Springer: Cham, Switzerland, 2014; Volume 24, pp. 1–18. [CrossRef]
26. Calida, B.Y.; Jaradat, R.M.; Abutabenjeh, S.; Keating, C.B. Governance in systems of systems: A systems-based model. *Int. J. Syst. Syst. Eng.* **2016**, *7*, 235–257. [CrossRef]
27. Katina, P.F.; Keating, C.B.; Bobo, J.A.; Toland, T.S. A Governance Perspective for System-of-Systems. *Systems* **2019**, *7*, 54. [CrossRef]
28. Schoenberg, M. Context in Complex Systems Governance. In *Complex System Governance*; Keating, C.B., Katina, P.F., Chesterman, C.W., Jr., Pyne, J.C., Eds.; Topics in Safety, Risk, Reliability and Quality; Springer: Cham, Switzerland, 2022; Volume 40, pp. 1–18. [CrossRef]
29. Mogahed, M.; Elahi, A.; Mansouri, M. Operationalizing Healthcare Through HHS Mission–Driven Analysis and Functional Categorization for a Governable Systems of Systems. In Proceedings of the 2025 IEEE International Symposium on Systems Engineering (ISSE), Paris, France, 28–30 October 2025.
30. Alsarhan, T.; Harfoushi, O.; Shdefat, A.Y.; Mostafa, N.; Alshinwan, M.; Ali, A. Improved Graph Convolutional Network with Enriched Graph Topology Representation for Skeleton-Based Action Recognition. *Electronics* **2023**, *12*, 879. [CrossRef]
31. Alsarhan, T.; Ali, S.S.; Ganapathi, I.I.; Ali, A.; Werghi, N. PH-GCN: Boosting Human Action Recognition Through Multi-Level Granularity with Pair-Wise Hyper GCN. *IEEE Access* **2024**, *12*, 162608–162621. [CrossRef]
32. Mogahed, M.; Mansouri, M. Enhancing Healthcare Delivery through Systems of Systems Governance: A Multi-Layered Governance Framework. In Proceedings of the 2025 INCOSE International Symposium, Ottawa, ON, Canada, 26–31 July 2025; Volume 35, pp. 1392–1405. [CrossRef]
33. Mogahed, M.; Mansouri, M. Advancing Sustainable Healthcare Delivery Through Systems of Systems Governance and Sustainability Governance Theories. In Proceedings of the 2025 20th Annual Systems of Systems Engineering Conference (SoSE), Tirana, Albania, 8–11 June 2025; pp. 1–6. [CrossRef]
34. Nygaard, K.; Dahl, O.-J. The development of the Simula languages. *ACM SIGPLAN Not.* **1978**, *13*, 245–272. [CrossRef]
35. Dahl, O.-J. The birth of object orientation: The Simula languages. In *From Object-Orientation to Formal Methods*; Owe, O., Krogdahl, S., Lyche, T., Eds.; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2004; Volume 2635, pp. 15–25. [CrossRef]
36. Sutherland, I.E. *Sketchpad: A Man–Machine Graphical Communication System*; Technical Report UCAM-CL-TR-574. University of Cambridge, Computer Lab.: Cambridge, UK, 2003. Available online: https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-574.pdf (accessed on 1 November 2025).
37. Kay, A. The early history of Smalltalk. In *History of Programming Languages II*; Bergin, T.J., Gibson, R.G., Eds.; ACM Press: New York, NY, USA, 1996; pp. 511–598. [CrossRef]

38. Kay, A.; (Hewlett-Packard Labs, Palo Alto, CA, USA); Ram, S.; (Freie Universität Berlin, Berlin, Germany). On the meaning of 'object-oriented programming'. Personal communication, July 2003.
39. Robson, D. Object-oriented software systems. *Byte Mag.* **1981**, *6*, 74–82.
40. Priestley, M. *A Science of Operations: Machines, Logic, and the Invention of Programming*; Springer: London, UK, 2011. [CrossRef]
41. Kafura, D.; Lavender, G. Concurrent object-oriented languages and the inheritance anomaly. In *Parallel Computers: Theory and Practice*; Quinn, M., Ed.; McGraw-Hill: New York, NY, USA, 1993; pp. 221–264.
42. Goetz, B. Fixing the Java Memory Model. *IBM DeveloperWorks*. 2004. Available online: https://web.archive.org/web/*/https://www.ibm.com/developerworks/java/library/j-jtp02244/ (accessed on 1 November 2025).
43. Milner, R. *Communicating and Mobile Systems: The π-Calculus*; Cambridge University Press: Cambridge, UK, 1999; ISBN 0-521-64320-1.
44. Joque, J. The invention of the object: Object orientation and the philosophical development of programming languages. *Philos. Technol.* **2016**, *29*, 335–356. [CrossRef]
45. Smith, B.C. *On the Origin of Objects*; MIT Press: Cambridge, MA, USA, 1996; ISBN 0262193639.
46. Barad, K. *Meeting the Universe Halfway: Quantum Physics and the Entanglement of Matter and Meaning*; Duke University Press: Durham, NC, USA, 2007. [CrossRef]
47. Carvajal, R.; Orellana, R.; Katselis, D.; Escárate, P.; Agüero, J.C. A systematic optimization approach for a class of statistical inference problems utilizing data augmentation. *arXiv* **2016**, arXiv:1601.05074.
48. Kramer, M.; Hubwieser, P.; Brinda, T. A Competency Structure Model of Object-Oriented Programming. In *2016 International Conference on Learning and Teaching in Computing and Engineering (LaTICE)*; IEEE: Mumbai, India, 2016; pp. 1–8. [CrossRef]
49. Li, M.; Bi, Z.; Wang, T.; Chen, K.; Xu, J.; Niu, Q.; Liu, J.; Peng, B.; Zhang, S.; Pan, X.; et al. Deep Learning and Machine Learning, Advancing Big Data Analytics and Management: Object-Oriented Programming. *arXiv* **2024**, arXiv:2409.19916. [CrossRef]
50. Mars, J. Data spatial programming: An OOP extension for topological models. *J. Syst. Archit.* **2025**, *140*, 103978. [CrossRef]
51. Liu, L.; Özsu, M.T. *Encyclopedia of Database Systems*, 1st ed.; Springer: New York, NY, USA, 2009. [CrossRef]
52. Bitter, R.; Mohiuddin, T.; Nawrocki, M. *LabVIEW: Advanced Programming Techniques*, 2nd ed.; CRC Press: Boca Raton, FL, USA, 2007; ISBN 9780849333255.
53. Azani, C. An open systems approach to system of systems engineering. In *System of Systems Engineering: Innovations for the 21st Century*; Jamshidi, M., Ed.; Wiley: Hoboken, NJ, USA, 2008; Chapter 2. [CrossRef]
54. National Electrical Manufacturers Association (NEMA). *Digital Imaging and Communications in Medicine (DICOM), NEMA PS3*; NEMA: Rosslyn, VA, USA, 2019. Available online: https://www.dicomstandard.org/ (accessed on 24 April 2025).
55. HL7 International. *FHIR Release 4 (v4.0.1)*; HL7 International: Ann Arbor, MI, USA, 2019; Available online: https://hl7.org/fhir/R4/ (accessed on 24 April 2025).
56. European Parliament and Council of the European Union. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data and repealing directive 95/46/EC (general data protection regulation). *Off. J. Eur. Union* **2016**, *L119*, 1–88. Available online: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32016R0679 (accessed on 1 November 2025).
57. European Health Data Space Regulation (EHDS). *Proposal for a Regulation of the European Parliament and of the Council on the European Health Data Space, COM(2022) 197 Final*; European Commission: Brussels, Belgium, 2022. Available online: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=celex:52022PC0197 (accessed on 1 November 2025).
58. Yin, R.K. *Case Study Research and Applications: Design and Methods*, 6th ed.; SAGE Publications: Thousand Oaks, CA, USA, 2018; Available online: https://uk.sagepub.com/en-gb/eur/case-study-research-and-applications/book250150 (accessed on 1 November 2025).
59. Flyvbjerg, B. Case Study. In *The Sage Handbook of Qualitative Research*, 4th ed.; Denzin, N.K., Lincoln, Y.S., Eds.; SAGE Publications: Thousand Oaks, CA, USA, 2011; pp. 301–316. Available online: https://www.academia.edu/3271072/Case_Study (accessed on 1 November 2025).
60. Elhauge, E.R. (Ed.) *The Fragmentation of U.S. Health Care: Causes and Solutions*; Oxford University Press: New York, NY, USA, 2010. [CrossRef]
61. Queen Mary University of London; BPE Solicitors LLP; Pinsent Masons LLP. *Data Trusts: Legal and Governance Considerations*; Report for the Open Data Institute; The Open Data Institute: London, UK, 2019; Available online: https://theodi.org/article/data-trusts-legal-report/ (accessed on 1 November 2025).
62. SymbolicData. Data Contracts: The Ultimate Guide to Implementation, Best Practices, and Real-World Examples. Available online: https://www.symbolicdata.org/data-contracts/ (accessed on 1 November 2025).
63. *ISO/TS 13131:2014*; Health Informatics—Telehealth Services—Quality Planning Guidelines. International Organization for Standardization (ISO): Geneva, Switzerland, 2014.
64. Health Level Seven International. *HL7 Version 2.9.0 Normative Standard*; Health Level Seven International: Ann Arbor, MI, USA, 2019. Available online: https://www.hl7.org/implement/standards/product_brief.cfm?product_id=185 (accessed on 24 April 2025).

65. World Health Organization. *ICD-11: International Classification of Diseases, 11th Revision*; WHO: Geneva, Switzerland, 2022; Available online: https://icd.who.int/ (accessed on 24 April 2025).
66. SNOMED International. *SNOMED CT*; SNOMED International: London, UK, 2025. Available online: https://www.snomed.org/ (accessed on 24 April 2025).
67. Regenstrief Institute. *LOINC: Logical Observation Identifiers Names and Codes*; Regenstrief Institute: Indianapolis, IN, USA, 1994; Available online: https://loinc.org/ (accessed on 24 April 2025).
68. Health Level Seven International. *CDA® R2 Implementation Guide: Consolidated CDA Templates for Clinical Notes—US Realm (C-CDA v2.1)*; Health Level Seven International: Ann Arbor, MI, USA, 2015; Available online: https://www.hl7.org/implement/standards/product_brief.cfm?product_id=492 (accessed on 24 April 2025).
69. Clinical Data Interchange Standards Consortium. *CDISC Standards*; CDISC: Austin, TX, USA, 2025. Available online: https://www.cdisc.org/standards (accessed on 24 April 2025).
70. Accredited Standards Committee X12. *ANSI ASC X12 Standards*; ASC X12: Falls Church, VA, USA, 2025. Available online: https://x12.org/ (accessed on 24 April 2025).
71. U.S. Food and Drug Administration. *Unique Device Identification System (UDI System)*; FDA: Silver Spring, MD, USA, 2018. Available online: https://www.fda.gov/medical-devices/device-advice-comprehensive-regulatory-assistance/unique-device-identification-system-udi-system (accessed on 24 April 2025).
72. EDICOM. Electronic Data Interchange (EDI); EDICOM Group: Valencia, Spain. Available online: https://edicomgroup.com/edi (accessed on 24 April 2025).
73. *ISO 7101:2023*; Healthcare Organization Management—Management Systems for Quality in Healthcare Organizations—Requirements. International Organization for Standardization (ISO): Geneva, Switzerland, 2023.
74. *ISO 13606-1:2019*; Electronic Health Record Communication—Part 1: Reference Model. International Organization for Standardization (ISO): Geneva, Switzerland, 2019.
75. *ISO 14155:2020*; Clinical Investigation of Medical Devices for Human Subjects—Good Clinical Practice. International Organization for Standardization (ISO): Geneva, Switzerland, 2020.
76. *ISO 45001:2018*; Occupational Health and Safety Management Systems—Requirements with Guidance for Use. International Organization for Standardization (ISO): Geneva, Switzerland, 2018.
77. U.S. Department of Health and Human Services. *Health Insurance Portability and Accountability Act of 1996 (HIPAA), Public Law 104-191*; U.S. Government Publishing Office: Washington, DC, USA, 1996. Available online: https://www.govinfo.gov/content/pkg/PLAW-104publ191/pdf/PLAW-104publ191.pdf (accessed on 24 April 2025).
78. *ISO/TC 215*; Health Informatics. International Organization for Standardization (ISO): Geneva, Switzerland, 2025. Available online: https://www.iso.org/committee/54960.html (accessed on 24 April 2025).

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.