

CORPORATION AS CODE

Rethinking Corporate Records for the Digital Age

Xavier Beauchamp-Tremblay & Frédéric Boivin Couillard

Draft — February 2026

Abstract

Corporate law remains conceptually tethered to paper. Charters, bylaws, resolutions, and minute books are treated as static, document-centric artifacts rather than as structured representations of legal relationships. This paper introduces the concept of the corporation as code: a model in which the legal entity is natively represented as a structured, version-controlled data object. Corporate attributes (ownership, governance rules, historical actions, and organizational relationships) are expressed as standardized data structures, with traditional documents functioning as human-readable views generated from an authoritative computational source. Drawing on software engineering practices such as version control and schema validation, we show how this approach enables automation, interoperability, and legal certainty that document-based workflows cannot achieve.

This PDF was generated from the version-controlled source at:

github.com/corporationascode/paper

TABLE OF CONTENTS

I. Introduction: Beyond the Paperless Corporation	5
The Paper Paradigm	5
Why This Matters Now	5
The Core Thesis	6
Scope and Structure.....	7
II. From Paper Artifacts to Data Structures	9
The Limitations of Document-Based Representation	9
What Data Structures Offer	10
Real-World Precedents.....	11
The Document as Interface	11
Transition Paths	12
III. Existing Approaches and Their Limits	14
Entity Management Software.....	14
Digital Minute Books and Document Management	15
Rules as Code and Computable Law	16
The Gap.....	17
IV. The Corporation as Code.....	18
The Corporate Data Model	18
Version Control for Corporate State	19
Complex Structures: A Multi-Jurisdictional Example.....	20
Signatures and Authorization	22
Prior Art and Influences	23
What This Is Not.....	24
V. Distinguishing DAOs and On-Chain Organizations.....	25
What DAOs Are	25
The Key Distinction: Representation vs. Execution	25
Why the Distinction Matters	25
The Conservative Reading.....	26
VI. Scaling Corporate Work: Founders, Agents, and the One-Person Unicorn.....	27
The Rise of Solo Founders.....	27
Corporate Scalability	27
AI Agents and Corporate Operations	28
The One-Person Unicorn.....	29
Adoption Dynamics	29
VII. Legal Architecture as a Layer of Enterprise Architecture	32

The Enterprise Architecture Paradigm	32
The Missing Legal Layer	32
Legal Architecture Defined.....	33
Policy-as-Code	34
Real-Time Compliance	34
Integration with Corporation as Code.....	35
VIII. Beyond AI as a Patch: Determinism, Structure, and Legal Certainty	37
The Efficiency Illusion	37
Why the Patch Is Costly	38
Determinism and Legal Certainty	39
AI as Operator, Not Interpreter	39
Generative AI as the Lawyer's Programming Interface	40
Structure Before Intelligence	42
IX. Interoperability, Standardization, and the Role of the State	43
Value Independent of Infrastructure	43
Government APIs as Accelerants	43
The U.S. SEC: Government-as-API at Scale	43
Variable Progress Elsewhere	44
International Standards Taking Hold	44
Regulatory Context: The EU Data Act.....	44
Why Government APIs Matter	45
A Pragmatic Path Forward	45
X. Risks, Objections, and Governance Concerns.....	46
Legal Status.....	46
Security	46
Complexity.....	46
Avoiding Technosolutionism.....	47
XI. Conclusion: Toward Computable Corporate Law.....	48
The Argument Restated.....	48
The Path Forward	48
Interdisciplinary Work	49
Taking Action.....	49
Closing Reflection	50
Bibliography	51

Despite decades of digitization, corporate law remains conceptually tethered to paper. Charters, bylaws, resolutions, and minute books are now drafted, signed, and stored electronically, yet they continue to be treated as static, document-centric artifacts rather than as structured representations of legal relationships. This paper argues that the persistence of the "paper paradigm" obscures the true nature of the corporation in a digital environment and forecloses possibilities for automation, interoperability, and legal certainty.

We introduce the concept of the corporation as code: a model in which the legal entity is natively represented as a structured, version-controlled data object rather than as a collection of documents. Under this approach, corporate attributes (ownership, governance rules, historical actions, and organizational relationships) are expressed as standardized data structures, with traditional documents functioning as human-readable views generated from an authoritative computational source. Drawing on software engineering practices such as version control and schema validation, we show how these techniques provide more accurate and auditable representations of corporate change than document-based workflows.

The paper situates this proposal within existing developments in legal technology and "rules as code" initiatives. We argue that current entity management tools replicate paper-based assumptions behind proprietary interfaces. Rules-as-code efforts, while valuable, face significant implementation and change management complexity given the scale of legislative and regulatory drafting processes. Corporation-as-code offers a more tractable entry point: new businesses can adopt structured representations from inception, avoiding legacy conversion entirely.

We examine how structured corporate data enables new organizational forms, particularly the rise of solo founders operating significantly capitalized businesses with minimal administrative overhead. We propose integrating legal architecture as a layer within enterprise architecture frameworks, enabling continuous compliance monitoring. We observe that using AI to interpret unstructured documents represents an inefficient detour, compensating for poor original data design rather than addressing the underlying problem. Conversely, generative AI coding assistants now enable lawyers to work directly with structured corporate data: describing transactions in natural language, generating scripts to effectuate complex multi-jurisdictional changes, and producing validation tests that verify the resulting corporate structure. This capability, unimaginable even five years ago, dissolves the traditional barrier between legal expertise and programmatic implementation. The paper examines how government registries can enable interoperability through APIs and standard data formats, building on existing implementations in several jurisdictions.

By reframing the corporation as a computable artifact, this paper advances a design-oriented vision of corporate law in which legal certainty and automation arise from how legal entities are modeled from inception.

I. Introduction: Beyond the Paperless Corporation

The legal profession has, by most measures, gone paperless. Law firms store documents in the cloud, courts accept electronic filings, and corporate transactions close without a single sheet of paper changing hands. Yet despite this surface-level digitization, corporate law remains conceptually tethered to paper. The artifacts that define corporate existence (articles of incorporation, bylaws, shareholder resolutions, minute books) are still conceived, drafted, and treated as documents. They happen to be stored as PDFs rather than in filing cabinets, but their fundamental nature has not changed. They emulate sheets of paper rather than represent information in a natively digital form.

This paper argues that the persistence of what we call the "paper paradigm" is no longer adequate. It obscures the true nature of the corporation in a digital environment and forecloses possibilities for automation, interoperability, and legal certainty that structured data representations would enable.

The Paper Paradigm

In contemporary legal practice, corporate reality is represented through a familiar set of artifacts: articles of incorporation, bylaws, board and shareholder resolutions, share certificates, minute books, contracts, and regulatory filings. These documents are treated as authoritative, static records. When lawyers speak of "the corporate record," they mean a collection of such documents, typically organized chronologically and stored, whether physically or electronically, as the definitive account of the corporation's existence and actions.

Digitization, as it has occurred in legal practice, has largely meant converting paper into electronic documents. A PDF is not a rethinking of the document; it is a digital replica of a printed page. Word processors produce files optimized for eventual printing or PDF conversion. Even "born digital" documents, those never intended to exist on paper, retain the formatting conventions, pagination, and visual structure of their paper ancestors. They are designed for human reading, not for computation.

The consequences of this approach are significant. Documents become the source of truth rather than views over underlying information. The relationships between corporate elements (shareholders and shares, directors and resolutions, subsidiaries and parents) remain implicit in prose rather than explicit in structure. Extracting information requires reading, interpreting, and often re-keying data that could have been structured from the start.

Why This Matters Now

The scale of the problem is considerable. Industry estimates suggest that approximately ninety percent of enterprise-generated data is unstructured, comprising emails, documents, images, and similar artifacts that lack a predefined schema.¹ Legal data, including contracts, corporate records, and regulatory filings, constitutes a substantial portion of this unstructured mass.

A note on terminology: this paper distinguishes between *unstructured data* (free-form text, images, scanned documents requiring interpretation to extract meaning),

¹ IBM, "Structured vs. Unstructured Data: What's the Difference?"
<https://www.ibm.com/think/topics/structured-vs-unstructured-data>

structured data (fixed schemas, relational databases, rigidly defined fields), and *semi-structured data* (flexible schemas, self-describing formats like JSON and XML). When we propose representing corporations as "structured data," we use the term broadly to encompass both structured and semi-structured formats. What unites these categories, and distinguishes them from unstructured data, is explicit organization amenable to programmatic processing. A JSON document describing corporate ownership is semi-structured in the technical sense, but it is structured in the sense that matters: its contents can be queried, validated, and transformed without human interpretation.

Every corporate transaction generates documents that, despite containing highly structured information (parties, dates, amounts, conditions), encode that information in prose paragraphs rather than data fields.

This creates a growing mismatch between what organizations need and what their legal infrastructure provides. Modern enterprises depend on data pipelines, APIs, and automated workflows. Yet their legal foundations remain locked in formats that resist integration. Compliance checks require manual review. Due diligence means reading hundreds of documents. Corporate restructurings demand painstaking reconciliation of inconsistent records across jurisdictions.

Increasingly, organizations turn to artificial intelligence to bridge this gap. Large language models can extract information from contracts, summarize board minutes, and flag potential issues in corporate records. But this approach, using sophisticated AI to interpret documents that were never designed to be machine-readable, is fundamentally compensatory. It deploys advanced tools to work around poor underlying data structures rather than addressing the structure itself.

The Core Thesis

At a conceptual level, a corporation is not a stack of documents. It is an abstract legal construct defined by a set of attributes (jurisdiction of incorporation, legal form, capital structure), relationships (shareholders, directors, officers, subsidiaries), rules (governance constraints, voting thresholds, transfer restrictions), and a history of state changes over time (incorporations, amendments, issuances, transfers, dissolutions). These characteristics are far more naturally represented as structured data than as prose documents.

Software architects will recognize a familiar pattern here: the Model-View-Controller (MVC) paradigm. In this frame, the corporation's underlying data (ownership, governance, history, metadata) constitutes the *Model*, the authoritative source of truth. Documents, certificates, and reports are *Views*, rendered representations formatted for specific audiences and purposes. The processes that update corporate state (board approvals, share transfers, regulatory filings) are *Controllers*, mediating between human decisions and state changes. The paper paradigm conflates these layers, treating the View (the document) as if it were the Model itself. Corporation-as-code restores the separation: the Model is structured state, Views are generated on demand, and Controllers implement permitted state transitions with an append-only event log and a clear, unidirectional flow from decision to record to representation.

The central claim of this paper is that we should reconceive the corporation as what we term "corporation as code": a model in which the legal entity is natively represented as a structured, version-controlled data object rather than as a collection of documents. Under this approach, corporate attributes and relationships are expressed

as standardized data structures (JSON schemas, for example), with traditional documents functioning as human-readable views generated from an authoritative computational source. Electronic signatures attach to data states or commits, not merely to rendered documents.

This is a conceptual shift more than a technological one. The tools to implement such a model largely exist. Entity management platforms already store corporate information as structured data behind their interfaces. Government registries maintain corporate records in databases. The missing element is recognition that the document is not the corporation, and that treating it as such imposes unnecessary constraints on what corporate law can achieve.

Scope and Structure

This paper develops the corporation-as-code thesis across eleven sections. Following this introduction, Section II examines the transition from paper artifacts to data structures, drawing on examples from corporate registries that have begun exposing information through APIs. Section III surveys existing approaches in legal technology and entity management, arguing that current tools replicate paper-based assumptions behind proprietary interfaces. Section IV presents the corporation-as-code model in detail, drawing analogies from software engineering practices such as version control and continuous integration. Section V distinguishes this proposal from decentralized autonomous organizations (DAOs), which execute organizational logic as code rather than merely representing it.

The paper then turns to implications. Section VI considers how structured corporate representations enable new organizational forms, particularly the emergence of highly capitalized companies with minimal staff. Section VII proposes integrating legal architecture as a layer within enterprise architecture frameworks. Section VIII examines why using AI to interpret unstructured legal documents represents an inefficient detour, arguing that structure should precede intelligence. Section IX examines the role of government registries and public APIs in enabling interoperability.

Section X addresses risks, objections, and governance concerns, including questions of access, complexity, and the relationship between human judgment and automated systems. Section XI concludes with reflections on the path toward computable corporate law.

The paper sources its examples from, and focuses on, the corporate law tradition in common law jurisdictions (the United States, United Kingdom, Canada, and Australia) while recognizing that the underlying arguments apply more broadly. The goal is conceptual: to articulate a shift in mental models from the corporation as document to the corporation as code. This paper does not propose detailed legal analysis of whether "pure" corporation-as-code artifacts would be accepted as regulatory filings or address evidentiary questions in litigation. Rather, it explores how structured data can transform corporate practice within existing legal frameworks.

A note on scope: while this paper focuses on corporations for clarity and concision, the analysis extends naturally to other organizational forms. Partnerships, limited liability companies, non-profit organizations, trusts, and investment funds all face similar challenges: governance structures encoded in documents, membership and ownership tracked in prose, compliance obligations scattered across contracts and filings. Private investment funds, in particular, may prove early adopters given their repetitive fund

formation processes, standardized terms, and dense webs of investor obligations. The principles developed here apply wherever organizational complexity meets document-centric infrastructure.

II. From Paper Artifacts to Data Structures

The document's privileged status in corporate law is not accidental. It reflects centuries of practice in which paper was the only durable, portable, and verifiable medium for recording transactions and commitments. A signed charter, a sealed certificate, a witnessed resolution: these artifacts served simultaneously as evidence, communication, and authorization. Legal formalities developed around physical objects. Signatures authenticated identity, seals indicated authority, witnesses attested to execution. When disputes arose, courts examined documents.

When digitization arrived, it preserved these conventions rather than questioning them. Electronic documents replicated the form of paper documents. Digital signatures authenticated electronic files in the same way wet signatures authenticated paper. The result was a legal system that remained conceptually organized around documents even as the underlying medium changed. The document retained its privileged status as the source of truth for corporate existence, even when the information it contained could have been represented in fundamentally different ways.

The Limitations of Document-Based Representation

Treating documents as the authoritative representation of corporate reality imposes significant constraints. Consider what a typical corporate record contains: articles of incorporation, bylaws, board resolutions, shareholder resolutions, share certificates, option agreements, employment contracts, regulatory filings. Each document contains structured information (names, dates, amounts, conditions, relationships) encoded in prose paragraphs.

The relationships between these elements remain implicit. A share certificate describes an ownership interest, but it does not link programmatically to a shareholder record, a share class definition, or a capitalization table. These aggregate relationships (who owns what percentage, which share classes carry which rights, how ownership has changed over time) exist in the minds of lawyers and administrators, or in separate spreadsheets maintained alongside the documents, but not in the documents themselves. Extracting the corporate structure from a minute book requires reading and interpretation, not querying.

Versioning presents similar challenges. When a corporation amends its bylaws, the result is a new document (amended and restated bylaws) rather than a tracked change to a single evolving object. Understanding the history of a provision requires comparing successive documents, often manually. There is no native "diff" showing what changed between versions, no commit history recording who made changes and when, no ability to roll back to a previous state.

Every downstream use of corporate information requires extraction. Due diligence teams read documents to populate data rooms. Compliance systems parse filings to check regulatory requirements. Cap table software imports information from certificates and agreements. Each extraction introduces opportunities for error, and the same information, repeated across multiple documents, is prone to drift and inconsistency.

Document-based representations also resist interoperability. There is no standard way to exchange corporate information between systems. Each entity management platform, each law firm, each registry maintains its own formats and conventions. Moving corporate data from one system to another typically means exporting

documents and re-entering information, a process that scales poorly and undermines the efficiency gains that digitization was supposed to provide.

What Data Structures Offer

Structured data representations address these limitations directly. A data structure defines a schema in advance: fields with names and types, constraints on valid values, relationships between entities. Rather than encoding information in prose, structured data makes it explicit and queryable.

Consider how a corporation might be represented as a data structure. A core entity (call it `Corporation`) would have attributes: legal name, jurisdiction of incorporation, formation date, status, registered agent. Related entities would represent shareholders, directors, officers, share classes, and resolutions. Relationships would be explicit: a shareholder owns a specified number of shares of a particular class; a director was appointed by a resolution on a given date; a subsidiary is wholly owned by its parent.

History becomes native to the representation. Each change to the corporate structure (a share issuance, a director resignation, a bylaw amendment) is recorded as a discrete event with a timestamp, an author, and a description of what changed. The current state of the corporation is the result of applying all historical events in sequence. Any previous state can be reconstructed by replaying events up to a given point. Differences between states can be computed automatically.

Consider the analogy to a software repository. At the core is a Git-like layer: once corporate records are represented as canonical, structured data, the corporation maintains an immutable, versioned record of corporate facts. Proposed actions can be modeled as branches, changes as signed commits, and effective outcomes as merges. This layer supports diffs, provenance, and "as-of" views without embedding assumptions about users or permissions. A director resignation, for example, becomes a signed, timestamped, and attributable commit that is atomic, traceable, and replayable, rather than a document circulated by email.

On top of this record sits a governance and integration layer, analogous to a modern Git hosting platform. This layer exposes APIs, workflows, and subscriptions that allow external systems and counterparties to rely directly on the underlying facts. A lender can validate current signing authority (and any applicable limits) without requesting officer certificates. A tax advisor can query an ownership snapshot as of a specific reporting date. Counterparties can subscribe to notifications for the particular facts they depend on, for instance, officer appointments, registered address, and ownership thresholds, rather than awaiting periodic filings.

Access control lives squarely in this outer layer. Role-based access control governs who may read, propose, approve, or rely on different parts of the corporate record. A corporate secretary may have write access to board composition and read access to shareholder records; a director may have approval authority for resolutions but read-only access to cap table details; an investor's counsel may see only their client's holdings and related agreements; an auditor may receive broad but time-limited read access. External parties can be granted scoped API roles: a bank verifying corporate authority sees current officers and signing powers, and nothing else. Because permissions are expressed as data rather than embedded in document-sharing conventions, they become auditable, enforceable, and consistent across systems.

Finally, validation shifts from downstream review to input-time enforcement. Schemas can require that a share issuance reference a valid class, that a director appointment be supported by an approved resolution, or that jurisdictions be drawn from a defined set. Errors that would otherwise propagate through documents and spreadsheets are caught at the moment of entry, before they harden into the corporate record.

Real-World Precedents

This approach is not hypothetical. Several jurisdictions and organizations have begun treating corporate information as structured data rather than document collections.

The United Kingdom's Companies House provides a REST API exposing company data in JSON format.² Developers can query for company profiles, filing histories, officer appointments, and persons with significant control. The data is available programmatically, described by OpenAPI specifications, and exchangeable across any system that can consume JSON. Companies House describes JSON as "an open, standard format for storing and exchanging data" that is "easier to use than XML and more compact." This is corporate information treated as data, not documents.

Microsoft's Common Data Model offers a standardized framework for describing business entities.³ A `model.json` file defines entities with attributes (name, data type, description), relationships between entities, and metadata about data provenance and schema compliance. The pattern is directly applicable to corporate data: entities for corporations, shareholders, directors; attributes for names, dates, jurisdictions; relationships for ownership, control, appointment. If enterprise software can standardize on common data models for customers, products, and transactions, there is no principled barrier to standardizing corporate legal structures.

The U.S. Securities and Exchange Commission provides perhaps the most compelling proof of concept. Since 2018, the SEC has mandated Inline XBRL for financial filings, a format that is simultaneously human-readable and machine-readable.⁴ Rather than requiring separate documents for humans (HTML) and machines (XBRL exhibits), Inline XBRL creates "a single document that is both human-readable and machine-readable." Data is embedded directly in the filing; users can click on any tagged data point to see contextual information, citations, and accounting guidance. The SEC's EDGAR system then exposes this data through free, open REST APIs (no authentication required) with updates available in real-time as filings are disseminated.⁵

These examples prove feasibility. What remains is extending structured approaches beyond isolated registries and financial filings to the full corporate record.

The Document as Interface

² UK Companies House, "API Overview," <https://developer.company-information.service.gov.uk/overview/>

³ Microsoft, "Common Data Model - model.json," <https://learn.microsoft.com/en-us/common-data-model/model.json>

⁴ SEC Release No. 33-10514, "Inline XBRL Filing of Tagged Data" (June 28, 2018). <https://www.sec.gov/data-research/structured-data/inline-xbrl>

⁵ SEC, "EDGAR Application Programming Interfaces," <https://www.sec.gov/search-filings/edgar-application-programming-interfaces>

Reconceiving the corporation as a data structure does not eliminate documents. It repositions them. Rather than serving as the authoritative source of corporate reality, documents become generated artifacts: human-readable views rendered from underlying data.

A board of directors still receives resolutions to review and approve. Shareholders still receive certificates evidencing their ownership. Regulators still receive filings in prescribed formats. Courts still examine documentary evidence. But these documents are outputs, not inputs. They are generated from the authoritative data structure, formatted for their intended audience, and reproducible in the sense that any point-in-time view can be regenerated from the versioned source. (This does not mean documents are deleted; retention policies still apply, and the version-controlled history preserves the complete audit trail of what was generated and when.) Even the most traditional board, accustomed to reviewing polished PDF resolutions with formal letterhead and proper signature blocks, can be accommodated: the data structure generates whatever view the audience requires.

This approach requires no change to existing law. Governments do not prescribe a specific technology or process for how corporations organize and maintain their internal records or how they produce regulatory filings or government mandated documents. A corporation that maintains its stock ledger as a JSON database, generates share certificates as needed, and submits filings in whatever format the registry requires is already operating within existing legal frameworks. The corporation-as-code model is a practice improvement, not a regulatory proposal.

Under this model, a PDF is analogous to a print stylesheet applied to structured data. The information exists in the data structure; the PDF is one possible rendering. Different renderings can serve different purposes: a summary for executives, a detailed record for auditors, a prescribed form for regulators.

Electronic signatures attach to data states rather than rendered pages. When a director approves a resolution, the signature authenticates a specific state of the corporate record (identifiable by a hash or commit identifier), not a particular PDF file. The audit trail shifts from tracking which documents were signed to tracking which state changes were authorized, by whom, and when.

Transition Paths

The shift from documents to data structures need not be all-or-nothing. Organizations can begin by treating structured data as authoritative for new transactions while maintaining legacy documents as historical records. A corporation formed today could be "born digital" in the fullest sense, with its initial state recorded as structured data and documents generated as needed. Legacy documents can be indexed, linked to structured records, and progressively superseded without requiring complete conversion.

The born-digital advantage is significant. This advantage compounds over time and may create competitive pressure for adoption. Consider two companies operating under similar economic constraints. One relies on document-centric corporate records, requiring manual reconciliation, bespoke legal work, and periodic audits to answer routine governance questions. The other maintains structured records that enable automated compliance checks, instant verification of ownership and authority, and repeatable responses to diligence and regulatory inquiries.

The operational difference translates directly into administrative overhead. Document-centric systems scale by adding people, professional services, and process complexity; structured systems scale by adding, and connecting, data. In environments where margins are constrained, this difference is not cosmetic. Reducing the labor and external coordination required to maintain corporate correctness can materially affect viability, particularly as organizational complexity increases. Over time, firms with structured corporate infrastructure benefit from compounding administrative leverage that legacy systems struggle to replicate.

The goal is to make structure the default and documents the exception, reversing the current relationship in which data is extracted from documents rather than documents generated from data.

III. Existing Approaches and Their Limits

The vision outlined in the previous sections is not without precedent. Legal technology has evolved considerably over the past two decades, and several categories of tools now address corporate data management, entity governance, and legal process automation. Yet these existing approaches, while useful, fall short of treating the corporation as a first-class software object. They digitize workflows without challenging the underlying document-centric paradigm, and they often introduce new problems (vendor lock-in, proprietary formats, siloed data) even as they solve old ones.

Entity Management Software

The most direct attempt to manage corporate information as data comes from entity management platforms. Products like Carta, Athennian, Diligent Entities, and similar tools offer centralized databases for tracking corporate structures, cap tables, officer appointments, and compliance deadlines. These platforms store information in structured form behind their interfaces, generating documents (stock certificates, board consents, annual filings) as outputs.

In principle, this sounds like the corporation-as-code model described earlier. In practice, the resemblance is superficial.

First, these tools replicate paper workflows behind graphical interfaces. The user experience often mirrors the traditional process: draft a resolution, circulate for signature, file with the state. The underlying data model exists to support document generation, not to serve as the authoritative representation of the corporation. The document remains the deliverable; the database is merely a convenience.

Second, the data models are proprietary. Each platform defines its own schema, its own entity relationships, its own conventions for representing corporate structures. There is no interoperability between systems. Moving from one platform to another requires exporting documents (often as PDFs) and re-entering information manually. The structured data, which ought to be the corporation's most portable asset, remains inside a vendor's system.

Third, practical friction accumulates. The total cost of document-based corporate management extends far beyond obvious line items. Subscription fees for entity management platforms compound annually. Migration between platforms requires consulting engagements to map data, validate completeness, and reconcile discrepancies. IT teams spend cycles maintaining integrations, managing access controls, and troubleshooting sync failures across vendor clouds. Legal fees accrue each time a lawyer must reconstruct corporate history from scattered documents, verify consistency between a cap table and underlying agreements, or prepare a data room for due diligence. Error costs are particularly insidious: a miskeyed share number, an overlooked amendment, a filing deadline missed because it lived in a spreadsheet rather than an integrated system. These errors surface at the worst moments (closings, audits, disputes) when the cost of correction is highest. Export capabilities, while improving, often produce lossy representations: flattened spreadsheets, PDF archives, or formats that cannot be meaningfully imported

elsewhere.⁶ While data portability is now a widespread statutory requirement (including under the GDPR), its limited scope and technical carve-outs rarely guarantee the functional continuity required by enterprise users. Consequently, bespoke negotiation of exit rights remains essential to ensure the transition of structured, semantically meaningful datasets. Legal practitioners advising on SaaS contracts routinely recommend negotiating data portability terms precisely because such rights vary by vendor.⁷

The corporation-as-code alternative offers a different value proposition. Text files and structured data formats cost nothing to store. Git repositories are free. A founder can manage corporate data in markdown or JSON, host it in their own infrastructure (or locally), back it up trivially, and refactor at will. There is no vendor dependency, no subscription fee, no uptime concerns beyond one's own systems, no mandatory interface. Privacy is straightforward: sensitive data can live in separate files with different access controls, or values can be obfuscated with simple key-value substitution. Digital signatures present some complexity, but hashing a data structure is free and public-key cryptography is open source.

One might object that this approach suits only technically sophisticated founders, not large corporations whose boards expect polished interfaces. But this objection misunderstands the model. The structured data is the backend; the interface can be whatever the audience requires. A board that expects beautifully formatted PDF resolutions with proper signature blocks receives exactly that, rendered from the authoritative data. A director signing through an established e-signature platform signs a document generated from structured records. The difference is that the PDF is derived from data rather than the data being extracted from PDFs. For the solo founder or small team, the tradeoff is compelling regardless. Paying for good technology is reasonable, but entity management platforms charge substantial fees for what a software engineer would recognize as a straightforward data management problem. The underlying "codebase" is not complex; the complexity lies in the interface and integrations, which may not justify the cost for organizations comfortable with structured text.

Digital Minute Books and Document Management

A different category of tools addresses document organization rather than entity modeling: digital minute books, document management systems, and contract lifecycle management software. These tools organize documents, track versions, manage approvals, and automate certain workflows. They improve on filing cabinets and email attachments, but they do not fundamentally reconceive the corporation.

A digital minute book is still a minute book. Documents are the atoms of the system. The software may index document contents, extract metadata, and provide search functionality, but it does not model the corporation as a structured entity. Relationships between documents remain implicit or are expressed through folder hierarchies and naming conventions rather than through explicit data linkages.

⁶ Opara-Martins, Justice, Reza Sahandi, and Feng Tian. "Critical analysis of vendor lock-in and its impact on cloud computing migration: a business perspective." *Journal of Cloud Computing* 5, no. 1 (2016): 4. <https://doi.org/10.1186/s13677-016-0054-z>

⁷ World Commerce & Contracting. "SaaS Contracting Guide." June 2023. <https://www.worldcc.com/Portals/IACCM/SaaS%20Contracting%20Guide.pdf>

Contract lifecycle management tools face similar limitations. They are designed to track contract status, deadlines, and approval workflows, but they treat each contract as an independent document rather than as a component of a larger corporate structure. A share purchase agreement is stored as a PDF, not as a set of structured data about share transfers that integrates with the cap table.

Rules as Code and Computable Law

A more ambitious tradition has emerged under the banner of "Rules as Code" (RaC). Pioneered in New Zealand and gaining attention through OECD publications and academic initiatives at MIT and Stanford, RaC proposes encoding legal rules in machine-executable form alongside their natural language expression.⁸ If legislation and regulations were available as code, the reasoning goes, compliance could be automated, policy implications could be tested, and legal services could be delivered at scale.

The intuition behind RaC is sound, and its goals overlap substantially with those of this paper. Both approaches seek to make law computable. Both recognize that natural language, while essential for human understanding, is poorly suited to automated processing. Both envision a future, to us inevitable, in which legal artifacts exist in dual representations: human-readable and machine-executable.

Yet RaC faces implementation and change management challenges that corporation-as-code largely avoids. Legal rules are inherently interpretive. Statutes contain ambiguous terms, competing interpretations, and exceptions that depend on context. Encoding such rules requires resolving ambiguities that legislatures deliberately left open or that courts have not yet addressed. More significantly, rules change through complex legislative and regulatory processes. Each amendment requires updating the encoded representation, validating consistency, and managing the transition. The scale of legislative and regulatory drafting, spread across multiple jurisdictions and subject areas, makes comprehensive rules-as-code implementation a formidable undertaking.

The rise of large language models has introduced a probabilistic alternative for handling ambiguous rules, but this introduces its own complications around auditability and legal certainty.

Corporation-as-code offers a more tractable entry point. Corporate data is primarily factual, not normative. The question "who owns how many shares of what class" does not admit multiple interpretations in the way that "reasonable care" or "material adverse change" might. The capital structure of a corporation is deterministic: shares were issued or they were not; directors were appointed or they were not; filings were made or they were not.

Crucially, new businesses can adopt corporation-as-code from inception. There is no legacy conversion, no change management for existing encoded rules, no political process to navigate. A founder incorporating a company today can represent that company as structured data from the start, building the corporate record correctly rather than retrofitting it later. This fresh-start advantage makes corporation-as-code

⁸ Morris, Jason. "Blawx: Rules as Code Demonstration." MIT Computational Law Report, August 14, 2020. <https://law.mit.edu/pub/blawxrulesascodedemonstration/release/1>

adoption fundamentally simpler than retrofitting entire legal codes into machine-executable form.

The competitive implications deserve emphasis. In some industries, legacy enterprises with document-centric infrastructure may find themselves structurally unable to compete with digital-native entrants. The operational overhead of maintaining and reconciling paper-paradigm records becomes a permanent cost disadvantage.

The Gap

What is missing, then, is not better tools within the existing paradigms but a different paradigm altogether. Entity management platforms treat data as a backend for document generation rather than as the authoritative corporate representation. Document management systems organize documents without modeling the corporation. Rules-as-code initiatives tackle the hardest problem (normative rules) while leaving the easier problem (factual corporate data) largely unaddressed.

The corporation-as-code proposal fills this gap. It does not require solving the deep problems of legal interpretation. It requires only that we recognize what entity management platforms already know but do not operationalize: the corporation is a data structure, and documents are views over that structure. Making this explicit, and building systems that treat the data structure (not the documents) as authoritative, is the next step.

IV. The Corporation as Code

Having surveyed the limitations of existing approaches, we now turn to the affirmative proposal: the corporation as code. This section develops the model in detail, drawing on concepts from software engineering to illustrate how corporate structures can be represented, versioned, and managed as structured data objects.

The core intuition is simple. A corporation, at any moment, exists in a particular state: it has a jurisdiction, a legal form, a set of shareholders with defined ownership interests, a board of directors, officers, bylaws, and a history of actions taken. This state changes over time through discrete events: shares are issued, directors resign, bylaws are amended, subsidiaries are formed. The challenge is to represent this evolving state in a form that is precise, auditable, interoperable, and amenable to automation.

Software engineers face analogous challenges daily. A codebase exists in a particular state at any moment and evolves through discrete changes. Version control systems like Git track every change, recording who made it, when, and why. The current state of the code is the result of applying all historical commits in sequence. Any previous state can be reconstructed. Differences between states can be computed and displayed. Multiple contributors can work on the same codebase, with systems managing conflicts and merging changes.

The corporation-as-code model applies these patterns to corporate governance.

The Corporate Data Model

At the foundation of the model is a structured representation of the corporation. Rather than a collection of documents, the corporation is defined by a set of interrelated data entities. A minimal schema might include:

Corporation: The root entity, containing core attributes.

```
{
  "id": "corp-001",
  "legal_name": "Acme Holdings Inc.",
  "jurisdiction": "Delaware",
  "formation_date": "2024-03-15",
  "status": "active",
  "registered_agent": {
    "name": "CT Corporation",
    "address": "1209 Orange Street, Wilmington, DE 19801"
  }
}
```

Share Classes: Definitions of the types of equity the corporation can issue.

```
{
  "id": "class-common",
  "name": "Common Stock",
  "authorized_shares": 10000000,
  "par_value": 0.0001,
  "voting_rights": true,
  "votes_per_share": 1
}
```

Shareholders: Records of who owns what.

```
{
  "id": "holder-001",
  "entity_type": "individual",
  "name": "Jane Founder",
  "holdings": [
    {
      "share_class": "class-common",
      "shares": 5000000,
      "acquisition_date": "2024-03-15",
      "acquisition_type": "founder_issuance"
    }
  ]
}
```

Directors and Officers: The humans who govern and operate the corporation.

```
{
  "id": "director-001",
  "name": "Jane Founder",
  "role": "director",
  "appointed_date": "2024-03-15",
  "appointed_by": "resolution-001",
  "status": "active"
}
```

Resolutions: Formal corporate actions.

```
{
  "id": "resolution-001",
  "type": "board_resolution",
  "date": "2024-03-15",
  "title": "Initial Organization",
  "actions": [
    "Adoption of bylaws",
    "Appointment of initial directors",
    "Authorization of share issuance"
  ],
  "approved_by": ["incorporator-001"],
  "status": "adopted"
}
```

This is illustrative, not prescriptive. The specific schema would vary based on jurisdiction, corporate form, and use case. The point is that these elements, which currently exist scattered across dozens of documents, can be represented as structured, interrelated data.

Version Control for Corporate State

The real power of the model emerges when we apply version control concepts. In Git, a repository's history is a sequence of commits, each representing a discrete change. A commit contains a description of the change, a reference to the author, a timestamp, and a pointer to the repository state after the change.

Corporate state evolves similarly. Each corporate action (a share issuance, a director appointment, a bylaw amendment) can be represented as a commit to the corporate

repository. The commit records what changed, who authorized it, when it occurred, and what the corporate state looked like afterward.

Consider a simple example: issuing shares to a new investor.

Before the transaction, the shareholder registry shows the founder holding 5,000,000 shares. The corporate record reflects the initial authorized capital and bylaws.

The transaction consists of a board resolution authorizing the issuance, an amendment to the share class if additional shares are authorized, the creation of a new shareholder record for the investor, and updates to the cap table.

The commit bundles these changes together with metadata:

```
{
  "commit_id": "abc123",
  "timestamp": "2024-06-15T14:30:00Z",
  "author": "legal@acme.com",
  "message": "Series A financing: issue 1,000,000 shares to Venture Fund I",
  "authorization": {
    "type": "board_resolution",
    "resolution_id": "resolution-005",
    "approved_by": ["director-001", "director-002"]
  },
  "changes": [
    {"entity": "shareholders", "action": "create", "id": "holder-002"},
    {"entity": "cap_table", "action": "update"}
  ],
  "signatures": [
    {"signer": "director-001", "signature": "...", "timestamp": "..."}
  ]
}
```

This commit becomes part of the permanent corporate history. It cannot be altered without detection. The corporate state at any point in time can be reconstructed by replaying commits. Differences between states can be computed automatically, showing exactly what changed and when.

Complex Structures: A Multi-Jurisdictional Example

The benefits of corporation-as-code become most apparent in complex corporate structures. Consider a holding company with subsidiaries across multiple jurisdictions: a Delaware parent, a UK operating subsidiary, a Canadian subsidiary, and a Singapore entity. The group's general counsel serves as a director on all four boards.

When the general counsel resigns, the traditional document-centric approach requires:

- Drafting resignation letters for each entity (four documents)
- Preparing board resolutions accepting the resignation (four documents, each following jurisdiction-specific conventions)
- Updating the minute books of each entity
- Filing notifications with each relevant registry (Delaware Division of Corporations, UK Companies House, Corporations Canada, ACRA in Singapore)
- Updating internal records, cap tables, and organizational charts

- Coordinating signature collection across potentially different time zones

Each step requires a lawyer familiar with the specific jurisdiction's requirements. The work is repetitive but not identical; each jurisdiction has its own forms, filing deadlines, and procedural requirements. Mistakes propagate. If the resignation date is recorded inconsistently across entities, the discrepancy may surface months later during due diligence.

In a corporation-as-code model, the same transaction becomes a programmatic operation. The corporate group is represented as a connected data structure:

```
# Representation of a corporate group
corporate_group = {
    "parent": {"id": "acme-delaware", "jurisdiction": "Delaware"},
    "subsidiaries": [
        {"id": "acme-uk", "jurisdiction": "UK", "parent_id": "acme-delaware"},
        {"id": "acme-canada", "jurisdiction": "Canada", "parent_id": "acme-delaware"},
        {"id": "acme-singapore", "jurisdiction": "Singapore", "parent_id": "acme-delaware"}
    ]
}

# The director serves on all boards
director = {
    "id": "director-gc-001",
    "name": "Jane Doe",
    "role": "director",
    "appointments": [
        {"entity_id": "acme-delaware", "appointed": "2022-01-15"},
        {"entity_id": "acme-uk", "appointed": "2022-02-01"},
        {"entity_id": "acme-canada", "appointed": "2022-02-15"},
        {"entity_id": "acme-singapore", "appointed": "2022-03-01"}
    ]
}
```

The resignation can be effectuated with a script that iterates across the corporate structure:

```
def resign_director_from_group(director_id, resignation_date, reason):
    """
    Resign a director from all entities in the corporate group.
    Returns a transaction bundle for review and approval.
    """
    director = get_director(director_id)
    transaction = TransactionBundle(
        description=f"Resignation of {director.name} from all group entities",
        effective_date=resignation_date
    )

    for appointment in director.active_appointments:
        entity = get_entity(appointment.entity_id)

        # Create resignation record and end the appointment
        resignation = DirectorResignation(
            director=director_id,
            entity_id=entity.id,
            effective_date=resignation_date,
            reason=reason
        )
```

```

    # Generate jurisdiction-appropriate resolution
    resolution = generate_resolution(
        entity=entity,
        action_type="accept_director_resignation",
        parameters=resignation,
        template=get_template(entity.jurisdiction, "director_resignation")
    )

    # Queue registry filing
    filing = generate_filing(
        registry=get_registry(entity.jurisdiction),
        filing_type="director_change",
        entity=entity,
        change=resignation
    )

    # Adds artifacts and updates appointment end_date atomically
    transaction.add(resignation, resolution, filing)

    # Validate the transaction
    validation_results = validate_transaction(transaction)
    if validation_results.has_errors():
        raise ValidationError(validation_results.errors)

    return transaction

```

The script generates all required artifacts: resignation records, board resolutions formatted for each jurisdiction, and registry filings in the appropriate formats. The lawyer reviews the transaction bundle, verifies that the generated documents are correct, and approves. Upon approval, the corporate state updates atomically. Where possible, filings are submitted to each registry programmatically (or queued for submission where APIs are not available).

The advantages compound with complexity. Adding a fifth subsidiary requires no additional logic; the script handles it automatically. Changing the resignation date requires updating a single parameter, not coordinating across four sets of documents. Validation rules catch inconsistencies before they propagate. The audit trail shows exactly what changed across all entities, who approved it, and when.

Signatures and Authorization

A critical question arises: how do electronic signatures fit into this model? In the document-centric paradigm, signatures attach to documents. A director signs a PDF resolution. A shareholder signs a stock purchase agreement. The signed document is evidence of authorization.

In the corporation-as-code model, signatures attach to data states. When a director approves a transaction, they sign a cryptographic hash of the commit or the resulting corporate state. This signature attests that the director authorized the specific changes recorded in the commit.

The advantages are significant. A signature on a data state is more precise than a signature on a rendered document. The signed object is unambiguous: it is the exact data structure, not a particular visual rendering. The signature can be verified programmatically. Multiple signers can sign the same state without needing to coordinate document versions.

Traditional documents remain available as needed. A board resolution can be rendered from the data structure and signed if required for filing or litigation. But the document is a derivative artifact, not the authoritative record. The authoritative record is the signed commit.

Prior Art and Influences

The corporation-as-code concept builds on several intellectual traditions and existing infrastructure.

The Legal Entity Identifier (LEI) system, coordinated by the Global Legal Entity Identifier Foundation (GLEIF), demonstrates that global corporate identification is achievable.⁹ Every LEI is a 20-character alphanumeric code based on ISO 17442 that uniquely identifies legal entities participating in financial transactions. The Global LEI Index is a free, public directory connecting each identifier to reference data including ownership structure: "who is who" and "who owns whom." Endorsed by the G20 and Financial Stability Board, the LEI proves that standardized corporate data can work across jurisdictions.

The Beneficial Ownership Data Standard (BODS), developed by Open Ownership, provides an open schema for publishing beneficial ownership information as structured, interoperable data.¹⁰ BODS defines how to represent entities, natural persons, and the relationships between them (shareholding, voting rights, board appointments, trust arrangements). The standard uses JSON, supports historical data (not just current state), and is already being adopted by multiple governments. BODS demonstrates that the hard design work of modeling corporate ownership as structured data has already been done.

The CommonAccord project, active since the early 2010s, pioneered the treatment of legal documents as "files in folders shared on GitHub."¹¹ CommonAccord stores legal templates and agreements as text files in git repositories, enabling version control, collaboration, and automation. While focused primarily on contracts rather than corporate structures, CommonAccord demonstrates that git-based workflows can apply to legal artifacts.

Corporation-as-code naturally extends to certain categories of contracts, particularly those that are essentially statements of corporate fact. Consider intercompany intellectual property licenses within a corporate group: Entity X of the group is authorized to use trademark Y in jurisdiction Z. This is a contract in legal form, but its substance is a factual relationship between corporate entities and intellectual property rights. Missing or incorrect IP licenses can create significant problems (trademark ownership disputes, tax complications, liability gaps). When such licenses are tracked as structured relationships within the corporate data model, queries like "which entities are licensed to use which marks in which territories?" become trivial. The contract generates from the data; the data is authoritative.

With apologies for the self-citation, one of this paper's co-authors discussed the parallel between legal drafting and software development in 2017, arguing that "drafting a contract is very much like 'programming' the relationship between the

⁹ GLEIF, "Introducing the Legal Entity Identifier (LEI)," <https://www.gleif.org/en/organizational-identity/introducing-the-legal-entity-identifier-lei>

¹⁰ Open Ownership, "Beneficial Ownership Data Standard," <https://standard.openownership.org/>

¹¹ CommonAccord, <https://www.commonaccord.org/>

parties" and predicting that "GitHub-like workflows will be a mainstay of at least some types of legal practices."¹² Corporation-as-code extends this intuition from contract drafting to corporate structure itself.

Stanford's Computable Contracts Initiative has developed methods for representing contract terms as "highly structured data" amenable to automated processing.¹³ The insight that contracts can be computable extends naturally to corporate governance: if contract terms can be structured data, so can bylaws, resolutions, and capitalization tables.

The distributed systems community has explored similar patterns. Append-only logs and event sourcing (recording state changes as a sequence of events rather than overwriting current state) are established techniques for building auditable, reproducible systems. The corporation-as-code model applies these patterns to the legal domain.

Together, these influences suggest a layered architecture: LEI provides the unique identifier, BODS provides the ownership schema, and corporation-as-code extends the model to governance, history, and operations.

What This Is Not

To avoid confusion, we should clarify what the corporation-as-code model does not claim.

It does not claim that corporations should execute themselves automatically. The corporation remains a legal fiction, enforced by courts and regulators, governed by humans exercising judgment. The model concerns representation, not execution. How the corporation is modeled and documented is distinct from how it makes decisions.

It does not claim that all corporate information can be perfectly structured. Some corporate artifacts (contracts with third parties, litigation files, informal communications) will remain unstructured or semi-structured. The model proposes that core governance elements (ownership, governance, formal actions) should be structured, not that every corporate document must be.

It does not require blockchain or cryptocurrency. While distributed ledgers offer one possible infrastructure for maintaining corporate records, the model is infrastructure-agnostic. A git repository hosted on private servers, a database with audit logging, or a public blockchain could each theoretically serve as the backend. The choice is an implementation detail, not a definitional feature.

The next section distinguishes the corporation-as-code model from a related but distinct concept: the decentralized autonomous organization.

¹² Beauchamp-Tremblay, Xavier. "GitHub Workflow and Legal Drafting." Slaw, April 21, 2017. <https://www.slaw.ca/2017/04/21/github-workflow-and-legal-drafting/>

¹³ Stanford CodeX, "Computable Contracts Initiative," <https://law.stanford.edu/codex-the-stanford-center-for-legal-informatics/projects/stanford-computable-contracts-initiative/>

V. Distinguishing DAOs and On-Chain Organizations

The phrase "corporation as code" may evoke comparisons to decentralized autonomous organizations (DAOs), a category of blockchain-based entities that have attracted considerable attention in recent years. The comparison is understandable but ultimately misleading. DAOs and the corporation-as-code model differ in fundamental ways, and conflating them obscures rather than clarifies the proposal.

What DAOs Are

A DAO is an organization whose operational logic is implemented in smart contracts deployed on a blockchain.¹⁴ Members typically hold governance tokens that grant voting rights. Proposals are submitted, voted upon, and (if approved) executed automatically by the smart contract code. Funds held by the DAO are released according to programmed rules. The blockchain serves as both infrastructure and enforcement mechanism.

The appeal of DAOs lies in their automation and transparency. Governance rules are visible in the code. Execution is deterministic: if the voting threshold is met, the action occurs. Human intermediaries are minimized. The organization, in a meaningful sense, runs itself.

This self-executing quality is precisely what distinguishes DAOs from the corporation-as-code model.

The Key Distinction: Representation vs. Execution

The corporation-as-code proposal concerns how corporations are *represented*, not how they *execute*.

A traditional corporation is a legal fiction. It exists because a legal system says it does. Its powers derive from statutes and common law. Its actions are taken by human agents (directors, officers) whose authority flows from the corporate record as established by formal resolutions, appointments, and other corporate acts, all subject to applicable law. Courts and regulators enforce the rules. The corporation does not execute itself; it is executed by humans operating within a legal framework.

The corporation-as-code model does not change this fundamental nature. It proposes that the corporation's state (ownership, governance, history) should be represented as structured data rather than unstructured documents. The representation is code-like, but the corporation remains a legal abstraction, enforced by external legal institutions, governed by humans making discretionary decisions.

A DAO, by contrast, attempts to make the organization itself execute as code. Governance decisions are not merely recorded; they are implemented automatically by smart contracts. The blockchain replaces (or supplements) external legal enforcement. The organization's behavior is constrained by what the code permits, not merely what the code describes.

Why the Distinction Matters

¹⁴ Harvard Law School Forum on Corporate Governance, "A Primer on DAOs" (2022), <https://corpgov.law.harvard.edu/2022/09/17/a-primer-on-daos/>

Conflating corporation-as-code with DAOs creates unnecessary obstacles to adoption. DAOs face significant legal uncertainty: most jurisdictions have not clarified their legal status or liability allocation.¹⁵ DAOs are also tied to blockchain infrastructure, while corporation-as-code is infrastructure-agnostic, implementable with conventional databases, git repositories, or any auditable system.

That said, the approaches are not mutually exclusive. Some DAOs have formed legal wrappers (Wyoming LLCs, Cayman foundations) to obtain legal personality while maintaining on-chain governance.¹⁶ A corporation represented as structured data could integrate with blockchain-based voting or tokenized equity if desired. The data model provides the interface; specific automation mechanisms are implementation choices.

The Conservative Reading

Readers skeptical of blockchain technology or DAO governance should note that the corporation-as-code proposal requires neither. It is, at heart, a conservative claim: that we should represent corporations more precisely, using structured data rather than unstructured documents. The techniques involved (schemas, version control, audit trails) are decades old and proven in other domains.

¹⁵ Columbia Business Law Review, "DAO Liability," <https://journals.library.columbia.edu/index.php/CBLR/announcement/view/564>

¹⁶ Journal of Corporate Finance, "DAO Governance" (2025), <https://www.sciencedirect.com/science/article/pii/S0929119925000021>

VI. Scaling Corporate Work: Founders, Agents, and the One-Person Unicorn

The corporation-as-code model offers clear benefits in precision, auditability, and interoperability. But its most compelling application may be in enabling a new category of organization: the highly capitalized company with minimal human staff.

The Rise of Solo Founders

For decades, conventional startup wisdom held that companies need co-founders. Y Combinator practically mandated them.¹⁷ Venture capitalists viewed solo founders with suspicion, interpreting a single founder as a sign that the entrepreneur could not convince anyone else to join their mission. The lone founder was a red flag.

The data suggests a shift.

According to an analysis of Carta data (Carta being a leading equity management platform) published by Solo Founders, solo founders accounted for 36.3% of new startups formed in the first half of 2025, up from 23.7% in 2019.¹⁸ It bears noting that the organization Solo Founders has a stake in promoting data showing this growth, and comprehensive data on this trend remains limited. Still, the trend has sparked discussion in founder communities, with many pointing to AI-powered tools as a possible enabler.¹⁹

If these figures are accurate, the trajectory is notable: 24.5% in 2020, 26.0% in 2021, 27.2% in 2022, 27.8% in 2023, 30.5% in 2024, and now 36.3%. More than a third of new venture-backed companies may now be founded by individuals working alone.

The question is why, and what it means for corporate infrastructure.

Corporate Scalability

Startups have long obsessed over "technical scalability": the ability to serve more users without proportionally increasing infrastructure costs. A well-architected software system can serve a million users as easily as a thousand. This leverage is fundamental to the economics of technology companies.

But technical scalability has a less glamorous counterpart: corporate scalability. As companies grow, so do their administrative burdens. Capitalization tables become complex. Regulatory filings multiply. Board meetings require documentation. Equity grants demand tracking. Compliance deadlines accumulate. Traditionally, managing this complexity required either dedicated staff or expensive professional services.

¹⁷ The conventional wisdom that startups need co-founders was widely discussed in the startup community. See, e.g., "Question re solo founders: 'A startup is too much work for one person,'" Hacker News, 2015. <https://news.ycombinator.com/item?id=9239322>

¹⁸ Dutilh, Ari. "Solo Founders in 2025: Why One-Third of All Startups Are Flying Solo." Solo Founders, December 30, 2025. <https://solofounders.com/blog/solo-founders-in-2025-why-one-third-of-all-startups-are-flying-solo>

¹⁹ The "one-person unicorn" concept generated extensive discussion on Hacker News when Sam Altman suggested a one-person billion-dollar company was imminent. See Devin Coldewey, "Sam Altman says the 'one-person unicorn' could soon be a reality," TechCrunch, January 7, 2025. <https://techcrunch.com/2025/01/07/sam-altman-says-the-one-person-unicorn-could-soon-be-a-reality/>

For solo founders, corporate administration presents a particular challenge. While they can certainly engage professional services, cost constraints often mean they remain heavily involved in the process to minimize expenses. A single person building a product cannot simultaneously manage a minute book, track vesting schedules, prepare board consents, and file annual reports. The result is either administrative neglect (creating problems for future fundraising or exits) or excessive time diverted from core business activities.

The corporation-as-code model addresses this problem directly. If corporate state is represented as structured data, administrative tasks become amenable to automation. Generating a cap table summary is a database query, not a document review exercise. Filing a state annual report can be triggered automatically from the corporate record. Tracking compliance deadlines becomes a matter of configuration rather than calendar management.

In other words, corporation-as-code enables corporate scalability: the ability to maintain institutional-grade governance without proportionally increasing administrative overhead.

AI Agents and Corporate Operations

The timing of the solo founder trend is not coincidental. It corresponds with dramatic advances in AI capabilities, particularly in agentic systems capable of executing complex, multi-step tasks autonomously.

Frontier models released in late 2025 represent a step change in what AI agents can accomplish.²⁰ These systems excel at autonomous, multi-step tasks requiring sustained reasoning. They achieve higher success rates on real-world software engineering benchmarks while using significantly fewer computational resources. They coordinate multiple tools, maintain context across extended operations, and resist adversarial manipulation better than their predecessors.

The relevance to corporate operations is direct. If AI agents can reliably maintain and modify complex codebases (navigating dependencies, running tests, fixing bugs, committing changes), they can perform analogous operations on corporate data structures. A corporate repository is, in many ways, simpler than a software repository. The schema is more constrained. The validation rules are clearer. The operations (issue shares, appoint director, file report) are more standardized.

Consider what becomes possible when corporate state is structured and AI agents are capable:

Routine administration can be delegated. The agent monitors the corporate record, identifies upcoming deadlines, prepares required filings, and presents them for human approval. The founder reviews and approves rather than researches and drafts.

Compliance monitoring becomes continuous. Rather than periodic audits, the agent continuously validates that the corporate state conforms to applicable rules. Missing signatures, lapsed registrations, or incomplete records trigger alerts before they become problems.

²⁰ Anthropic, "Introducing Claude Opus 4.5," <https://www.anthropic.com/news/claude-opus-4-5>

Transaction preparation accelerates. When a financing round approaches, the agent assembles the required information from the corporate record, identifies discrepancies that need resolution, and prepares documentation in the formats investors and their counsel expect.

Governance documentation is generated on demand. Board minutes, written consents, and shareholder resolutions can be drafted from structured inputs describing the actions taken. The human reviews and signs; the agent handles formatting and filing.

None of this requires artificial general intelligence or science fiction scenarios. It requires only that corporate information be structured (enabling programmatic access) and that AI agents be capable of multi-step tasks with appropriate oversight (which current systems demonstrate).

The One-Person Unicorn

The logical endpoint of these trends is the "one-person unicorn": a billion-dollar company operated by a single founder, supported by AI agents and external service providers, with minimal or no traditional employees.

This is not purely speculative. We already see venture-backed companies with valuations in the hundreds of millions operated by teams of fewer than ten people. As AI capabilities expand and corporate administration becomes more automatable, the minimum viable team size continues to shrink.

To be clear: even a solo founder will likely engage outside counsel for significant transactions, complex governance questions, and regulatory filings that require professional judgment. The one-person unicorn does not mean the one-person legal department. What changes is the interface between founder and counsel. Instead of exchanging documents back and forth, counsel operates on the same structured corporate repository. The lawyer reviews the data model, validates proposed transactions against it, and commits changes that the founder approves. The founder is not reading minute books; neither is the lawyer recreating corporate history from scattered PDFs.

For such organizations, document-centric corporate infrastructure creates unnecessary friction. A solo founder cannot spend hours managing PDFs and coordinating signatures across multiple platforms. Paper-emulating workflows are incompatible with the operating model these founders are building, even when professional services handle the details.

Corporation-as-code offers an alternative. From incorporation, the corporate state exists as structured data. Lawyers, accountants, and AI agents all operate on the same authoritative record. Documents are generated when needed but are not the source of truth. The founder's attention remains on the business; professional advisors access a clean, queryable corporate record rather than reconstructing it from documents each time.

This vision is not limited to solo founders. Any organization seeking to minimize administrative overhead while maintaining governance rigor can benefit. But the solo founder represents the limiting case: if corporation-as-code can work for a single person running a substantial business, it can work for organizations of any size.

Adoption Dynamics

Retrofitting existing corporations into a code-native model presents challenges. Legacy organizations have accumulated years of document-based records. Converting this history to structured data requires effort that may not be justified for mature companies with established administrative processes.

New corporations face no such barrier. A company formed today can be born digital in the fullest sense. Its initial state can be recorded as structured data. Every subsequent change can be captured as a commit to the corporate repository. Documents can be generated as needed for regulatory filings or third-party interactions, but the structured record remains authoritative.

Competition may accelerate adoption, but the dynamics are more consequential than mere efficiency gains suggest. Consider the economics of a venture-backed technology company: constant financing rounds, option grants, refreshes, restructures, heavy board cadence, investor governance requirements. Each transaction requires documentation, due diligence, and coordination. In a document-centric model, each transaction consumes legal fees, management attention, and time. In a corporation-as-code model, transaction preparation becomes a data query; due diligence becomes API access; closing becomes a commit.

The velocity difference is substantial. A startup with structured corporate records can close a financing round in a fraction of the time required by document-centric competitors. They can respond to acquisition inquiries with instant data room generation. They can onboard investors without weeks of document assembly. Faster fundraising translates directly to competitive advantage: a company that closes rounds quickly can deploy capital sooner, scale ARR faster, and reach defensible positions before slower competitors.

The impact extends beyond individual transactions. Reduced administrative overhead means resources can be redeployed to higher-ROI work. Lower fixed costs reduce liquidity risk and improve operating leverage. Faster transactions reduce the friction costs of consolidation and restructuring. By compressing diligence and closing overhead, structured corporate records lower the threshold at which solo-founder ventures become viable.

Private investment funds represent another particularly compelling early adoption case. Fund formation involves repetitive processes: similar limited partnership agreements, standardized terms, predictable capital call and distribution mechanics, and dense webs of investor-specific obligations (side letters, most-favored-nation provisions, co-investment rights). A fund administrator managing dozens of funds with hundreds of limited partners faces exactly the kind of complexity that structured data handles well. Fund terms are a particularly clear case because they generate recurring, exception-heavy administration. When rights and elections (e.g., MFNs, reporting elections, fee and expense terms, consent thresholds) are represented as structured data rather than prose, sponsors can query obligations across the LP base, drive notices and elections from the record itself, and reconcile outcomes directly into fund administration and accounting workflows. The result is not just fewer errors, it is a lower marginal cost of operating the fund, compressing the GP's administrative burden and the fees required to support it.

The adoption path, then, likely begins with new formations rather than legacy conversions, with private investment funds and venture-backed startups as natural beachheads. As more founders and fund managers choose code-native infrastructure, the ecosystem of supporting tools, service providers, and regulatory accommodations

will develop. Over time, the benefits may become compelling enough to justify conversion even for established organizations.

The solo founder trend suggests demand exists. The AI agent trend suggests capability is emerging. Corporation-as-code provides the conceptual framework connecting them.

VII. Legal Architecture as a Layer of Enterprise Architecture

Modern organizations maintain elaborate maps of their structure. Enterprise architecture frameworks describe business units, processes, and capabilities. IT architecture describes systems, data flows, and infrastructure. These architectures are documented, version-controlled, and used for planning and impact analysis. When a business process changes, the enterprise architecture helps identify affected systems. When an IT system is modified, the architecture reveals downstream dependencies.

What is conspicuously absent from most enterprise architecture frameworks is a legal layer. Legal obligations attach to business units and information systems, but these attachments remain implicit, documented (if at all) in scattered policy documents, contracts, and compliance checklists. The corporation-as-code model enables a different approach: legal architecture as an explicit, structured layer integrated with enterprise and IT architecture.

The Enterprise Architecture Paradigm

Enterprise architecture as a discipline emerged to manage organizational complexity. Frameworks like TOGAF (The Open Group Architecture Framework) provide structured approaches for describing how an organization's business, data, application, and technology components fit together.²¹ These frameworks enable understanding how changes propagate across domains, identifying redundancies, and maintaining alignment between strategy and implementation.

A well-maintained enterprise architecture answers questions like "which systems need integration when we acquire a subsidiary?" through explicit encoded relationships rather than tribal knowledge.

This paradigm is well-established in the technology domain. Organizations invest significantly in maintaining architectural documentation, employing architects who understand both business and technical concerns, and tooling that visualizes dependencies and supports impact analysis. The return on this investment comes from faster decision-making, fewer integration failures, and better alignment between strategy and execution.

The Missing Legal Layer

Legal obligations pervade organizations but rarely appear in architectural representations. Consider a typical enterprise:

- The human resources business unit is subject to employment law, benefits regulations, and data protection requirements.
- The customer data system holds information subject to privacy regulations (GDPR, CCPA) and potentially industry-specific rules (HIPAA, PCI-DSS).
- The corporate entity structure includes subsidiaries in multiple jurisdictions, each with its own corporate law, tax obligations, and reporting requirements.
- Commercial contracts impose performance obligations, limitations of liability, and intellectual property constraints.

²¹ The Open Group, "TOGAF Standard," <https://www.opengroup.org/togaf>

These legal attachments are real and consequential. A change in privacy regulation affects specific systems. A corporate restructuring triggers filings in multiple jurisdictions. A new product launch requires compliance review across multiple legal domains.

Yet these relationships are rarely modeled with the same rigor as technical dependencies. Legal obligations exist in documents (contracts, policies, regulatory summaries) rather than in structured representations. Impact analysis requires lawyers to read documents and exercise judgment rather than querying a model. The result is slower response to legal changes, higher compliance costs, and increased risk of overlooking obligations.

The maturity gap is striking. Organizations have embraced Infrastructure as Code for technical systems, GitOps for deployments, and Policy as Code for security controls.²² Yet organizational structure, governance rules, and legal obligations (arguably more consequential than infrastructure configuration) remain trapped in static documents. The tools and paradigms exist; they have simply not been applied to this domain.

Legal Architecture Defined

A legal architecture layer would model legal obligations as structured entities with explicit relationships to business units, systems, and data. The core elements might include:

Legal Regimes: Structured representations of applicable law. A regime might describe a jurisdiction (Delaware corporate law), a regulatory domain (SEC reporting requirements), or a contractual framework (master services agreement with a key customer). Each regime specifies obligations, constraints, and compliance requirements.

Attachments: Explicit links between legal regimes and organizational elements. An attachment might state that the customer database is subject to GDPR, that the UK subsidiary is governed by Companies Act 2006, or that the sales process must comply with the terms of a distribution agreement.

Compliance States: Structured representations of the organization's compliance posture. For each attachment, the model records whether obligations are met, what evidence supports compliance, and what actions are required.

When legal requirements change (a new regulation, an amended contract, a court decision), the legal architecture enables immediate impact analysis. Which systems are affected? Which business units bear new obligations? What compliance actions are required? The answers emerge from querying the model rather than from document review.

A useful distinction here is between what we might call *legal data primitives* and interpretive legal questions. Legal data primitives are deterministic facts that can be verified programmatically: "Is this entity registered in Delaware?" "Does this system process personal data?" "Has this director's term expired?" "Does this investor have MFN rights?" These are binary questions with objective answers. Interpretive legal questions, by contrast, require judgment: "Does this conduct constitute a breach of

²² Rothmann, Daniel, "Company as Code," 42 Futures, February 25, 2025, <https://blog.42futures.com/p/company-as-code>

fiduciary duty?" "Is this disclosure materially misleading?" "Does this hybrid waterfall structure qualify as American or European?"

The legal architecture approach focuses on the former while acknowledging a continuum between the two categories. At one end are purely factual checks (consent exists or it does not; a filing was made or it was not). At the other end are questions requiring substantive legal analysis. But the boundary between these categories evolves. Over time, as market practice standardizes legal language and as AI-driven tools create pressure toward consistent interpretation, more questions migrate from the interpretive toward the deterministic end of the spectrum. Unique identifiers for contract clauses, standardized term definitions, and machine-readable legal language push interpretation toward convergence. In this light, legal data primitives become foundational: they are the core around which increasingly deterministic legal analysis accretes.

Policy-as-Code

Tools like Open Policy Agent (OPA) treat operational policies as executable specifications that can be version-controlled, tested, and audited like any other software artifact.²³

The benefits are significant: consistency across systems, centralized policy management, audit trails for policy changes, and the ability to test policies before deployment. The same policy can govern access to Kubernetes clusters, API endpoints, and database records.

Legal architecture extends this paradigm. Governance rules, compliance requirements, and regulatory constraints can be expressed as structured specifications. Systems can query the legal architecture to determine applicable obligations. Changes to legal requirements propagate automatically to affected systems.

Consider a concrete example: a privacy regulation requires explicit consent for processing personal data for marketing purposes. In a policy-as-code model, this requirement is expressed as a rule; the marketing system queries the policy engine before sending communications; non-compliant actions are blocked. The example works because checking whether consent exists is a factual query (a legal data primitive), even if determining what constitutes valid consent may require legal analysis. Organizations already implement such systems for technical policies. Extending the approach to legal requirements is a natural progression.

Real-Time Compliance

The integration of legal architecture with enterprise architecture enables a shift from periodic compliance review to continuous compliance monitoring. Rather than conducting annual audits that sample documents and interview stakeholders, organizations can maintain real-time visibility into their compliance posture.

The model knows which regulations apply to which systems. It knows the current state of corporate governance (directors, officers, filings). It knows the terms of material contracts. When something changes (a new regulation is enacted, a director resigns, a contract is amended), the impact is immediately visible. Affected obligations are

²³ Open Policy Agent Documentation, <https://www.openpolicyagent.org/docs>

flagged. Required actions are identified. Compliance dashboards update automatically.

This does not eliminate the need for legal judgment. Complex regulatory questions still require human analysis. Novel situations still require interpretation. But routine compliance monitoring, the work of tracking deadlines, verifying documentation, and maintaining records, becomes systematic rather than manual.

Integration with Corporation as Code

The legal architecture concept integrates naturally with corporation-as-code. The corporate data structure provides the foundation: entities, ownership, governance, history. The legal architecture layer adds the legal overlay: which regimes apply, what obligations exist, what compliance state obtains.

Consider how an integrated model might be structured:

```
# Corporate structure (from corporation-as-code)
corporate_group = {
    "parent": {
        "id": "acme-delaware",
        "jurisdiction": "US-DE",
        "systems": ["crm-main", "erp-global"]
    },
    "subsidiaries": [
        {"id": "acme-uk", "jurisdiction": "GB", "systems": ["crm-eu", "hr-uk"]},
        {"id": "acme-india", "jurisdiction": "IN", "systems": ["dev-bangalore"]},
        {"id": "acme-brazil", "jurisdiction": "BR", "systems": ["sales-latam"]}
    ]
}

# Technology architecture
systems = {
    "crm-eu": {"data_types": ["customer_pii", "contact_info"], "location": "eu-west"},
    "hr-uk": {"data_types": ["employee_pii", "payroll"], "location": "eu-west"},
    "dev-bangalore": {"data_types": ["customer_pii", "analytics"], "location": "ap-south"},
    "sales-latam": {"data_types": ["customer_pii", "transactions"], "location": "sa-east"}
}

# Legal architecture layer
legal_regimes = {
    "gdpr": {
        "applies_to": ["customer_pii", "employee_pii"],
        "jurisdictions_with_adequacy": ["US-DE", "GB", "CA", "JP"], # simplified
        "requires_for_transfer": "adequacy_decision OR standard_clauses OR
binding_rules"
    }
}

def find_gdpr_transfer_risks(corporate_group, systems, legal_regimes):
    """
    Find all systems processing personal data for entities in
    jurisdictions without GDPR adequacy decisions.
    """
    risks = []
    gdpr = legal_regimes["gdpr"]
    all_entities = [corporate_group["parent"]] + corporate_group["subsidiaries"]

    for entity in all_entities:
```

```

        if entity["jurisdiction"] not in gdpr["jurisdictions_with_adequacy"]:
            for system_id in entity.get("systems", []):
                system = systems.get(system_id, {})
                pii_types = [d for d in system.get("data_types", [])
                             if d in gdpr["applies_to"]]
                if pii_types:
                    risks.append({
                        "entity": entity["id"],
                        "jurisdiction": entity["jurisdiction"],
                        "system": system_id,
                        "data_types": pii_types,
                        "required_safeguard": gdpr["requires_for_transfer"]
                    })
    return risks

# Execute the query
risks = find_gdpr_transfer_risks(corporate_group, systems, legal_regimes)
# Returns:
# [
#   {"entity": "acme-india", "jurisdiction": "IN", "system": "dev-bangalore",
#    "data_types": ["customer_pii"], "required_safeguard": "adequacy_decision OR..."},
#   {"entity": "acme-brazil", "jurisdiction": "BR", "system": "sales-latam",
#    "data_types": ["customer_pii"], "required_safeguard": "adequacy_decision OR..."}
# ]

```

This query, which would require hours of document review and legal analysis in a traditional setting, executes in seconds. Note that this query asks factual questions (which entities exist, what data they process, which jurisdictions lack adequacy decisions) rather than encoding the substantive legal rules themselves. The legal architecture models facts and relationships; human judgment determines what those facts mean. The answer combines IT architecture (which systems exist and what data they process), corporate structure (which subsidiaries operate in which jurisdictions), and legal architecture (which privacy rules apply and what transfer mechanisms are required). No single document contains this information. The integrated model does.

The same pattern applies to other compliance questions:

- "Which entities have directors whose terms expire in the next 90 days?"
- "What filings are required if we change our registered agent in Delaware?"
- "Which contracts reference the subsidiary we're planning to dissolve?"

Each question spans multiple architectural layers. Each would traditionally require a lawyer to read documents, consult spreadsheets, and exercise judgment. With an integrated model, each becomes a query returning precise, auditable results.

The result is legal reasoning that is architectural rather than artisanal. Compliance becomes a structural property of how the organization is modeled, not an after-the-fact review of whether documents meet requirements.

VIII. Beyond AI as a Patch: Determinism, Structure, and Legal Certainty

The legal industry has embraced artificial intelligence with considerable and appropriate enthusiasm. Large language models can review contracts, summarize documents, draft correspondence, and answer legal research questions with impressive fluency. AI-powered due diligence tools can process thousands of documents in seconds or minutes. Contract analysis platforms extract key terms and flag risks at scale. These capabilities are genuinely valuable, and the technology continues to improve.

The temptation, then, is to view AI as the solution to the problems described in this paper: if corporate documents are unstructured, use models to interpret them. Upload the minute book to a language model. Ask it questions. Let the AI extract the cap table, identify the current directors, summarize the board resolutions.

For the specific problem of corporate data, this approach is a Rube Goldberg machine: an elaborate workaround for a problem that admits a simpler solution.

The Efficiency Illusion

Consider what happens when an organization uses AI to manage unstructured corporate records. The minute book is a collection of PDFs. The cap table is a spreadsheet that may or may not match the stock purchase agreements. The board resolutions are scattered across email attachments and document management systems. To answer a simple question ("How many shares does Investor X hold?"), the AI must:

1. Identify which documents might contain the answer
2. Parse the documents (handling varied formats, scanned images, inconsistent naming)
3. Extract the relevant information
4. Reconcile conflicts between sources
5. Present an answer with appropriate caveats

This works, more or less. Modern AI systems are remarkably capable at document interpretation. Hallucination risks, while not eliminated, have diminished substantially for sophisticated tools that ground responses in source documents. The outputs require verification, but a well designed process absolutely saves time compared to manual review.

Yet practitioners at the frontier of AI-powered document analysis report that the challenges are worse than they first appear. Foundation-model capability is no longer the main bottleneck for extraction. With tuned prompts, the right model choice, and a well-designed retrieval pipeline, 92%+ accuracy is already being delivered, and performance improves as models and retrieval quality advance.²⁴ The complexity comes from handling long-tail prompting edge cases to reach higher accuracy, and from operating reliably under degraded or adversarial document conditions. Virtual data rooms, e-signature platforms, and other systems increasingly deploy encryption,

²⁴ OpenAI, "Hebbia's Deep Research Automates 90% of Finance and Legal Work, Powered by OpenAI," <https://openai.com/index/hebbia/>

watermarking, and obfuscation techniques that actively prevent AI analysis. Parsing such documents requires complex parsing pipelines to handle handwritten annotations, table structure extraction, visual overlays (such as watermarks), and other obfuscating artifacts, and to contend with restricted or encrypted files when access is authorized. These pipelines typically become more complex and costly as these techniques become more sophisticated.

Even when documents are accessible, the interpretive work does not disappear; it is encoded into prompting infrastructure. Consider the task of classifying the distribution waterfall structure from a limited partnership agreement. Hybrid structures often blur the line between "American" and "European" distribution mechanics, forcing practitioners to make judgment calls about classification. If you want to turn legal documents into data primitives, you must map this interpretive process into a system that can consistently process variations across thousands of documents. The work is possible but hard, expensive, and inherently subjective at the margins.

Most "hallucinations" in sophisticated production systems are not model hallucinations in the technical sense. They are prompt-specification issues, including prompt-induced bias and instructions that appear unambiguous yet leave edge-case behavior underconstrained.

The problem, then, is not that AI fails. The problem is that the entire exercise is unnecessary overhead. We are deploying expensive computational resources to recover structure that should have been present from the start. We are asking AI to clean up a mess rather than preventing the mess.

Why the Patch Is Costly

Using AI to interpret unstructured corporate data imposes several costs beyond the obvious computational expense:

Data spreads rather than consolidates. Each AI tool maintains its own index, its own cache, its own interpretation of the source documents. The corporate data that should live in one authoritative place instead exists in fragments across multiple vendor systems. Switching tools means re-ingesting documents and hoping for consistent interpretation. Worse, model deprecations and updates can change behavior in previously stable extraction pipelines. Organizations therefore incur ongoing costs not only to build these systems, but to continuously monitor, validate, and re-tune them as models evolve, costs that scale poorly with adoption.

The underlying mess persists. AI interpretation does not fix bad data; it routes around it. The minute book remains disorganized. The cap table spreadsheet remains out of sync with the legal documents. The next time someone needs to answer a corporate governance question, they face the same unstructured mess, perhaps with a different AI tool that interprets it slightly differently.

Verification overhead remains. For consequential decisions (financing rounds, M&A due diligence, regulatory filings), professionals must verify AI outputs against source documents. The AI provides a first pass, but human review catches the cases where interpretation went wrong. This is appropriate caution, but it means the efficiency gains are smaller than they first appear.

Bad habits compound. When AI makes unstructured data tolerable, the incentive to structure data properly diminishes. Organizations continue generating documents

in formats that require interpretation rather than formats that encode information directly. The problem perpetuates itself.

Compliance burden multiplies. The costs extend beyond technical infrastructure. Rothmann (2025), proposing "Company as Code" as a structured representation of organizational policies and operations, describes spending hundreds of person-hours collecting evidence and updating documentation for ISO 27001 audits, work that could have been invested in product development.²⁵ While Rothmann focuses on the organizational layer (people, roles, operational policies), the compliance overhead he identifies applies equally to the legal entity foundation we address. With policy-as-code and structured organizational data, auditors could query the organizational model directly, with traceable mappings from policy statements to implementation evidence, rather than requiring manual evidence gathering for each audit cycle. The overhead of maintaining dual representations (operational reality in systems, compliance documentation in documents) compounds as regulatory requirements expand.

Determinism and Legal Certainty

Legal systems value certainty. Property rights, contract enforcement, and corporate governance depend on clear answers to deterministic questions. Ambiguity exists in law (what constitutes "reasonable" behavior, whether a contract term is enforceable), but much corporate information is not inherently ambiguous. The capital structure of a corporation is deterministic and objective. The identity of directors is deterministic. The filing history with state regulators is deterministic.

These deterministic elements should be represented in ways that yield consistent answers when queried. A structured database returns the same result for the same query. A JSON schema either validates or fails. A git commit history shows exactly what changed and when. There is no interpretation, no probability, no hallucination risk.

The corporation-as-code model achieves this determinism by design. Corporate state is structured data. Queries return deterministic answers, not interpretations. Changes are logged immutably. The history is complete and auditable.

This does not eliminate all ambiguity from corporate law. Interpretive questions remain: What does a bylaw provision mean? Did a board act within its authority? Is a transaction fair to minority shareholders? These questions require human judgment and may ultimately require court resolution. But the deterministic substrate on which these questions depend (who owned what, what was authorized, what was disclosed) should not itself be a source of uncertainty.

AI as Operator, Not Interpreter

The appropriate role for AI in corporation-as-code is not as interpreter of unstructured documents but as operator on structured data.

An AI agent managing a corporate repository does not read PDFs and guess at their meaning. It queries structured records, validates inputs against schemas, generates documents from templates, and triggers workflows based on defined rules. When it

²⁵ Rothmann, Daniel, "Company as Code," 42 Futures, February 25, 2025, <https://blog.42futures.com/p/company-as-code>

encounters ambiguity (a missing field, a constraint violation, an unusual request), it escalates to humans rather than guessing.

This is a fundamentally different posture. The AI operates within a deterministic framework, performing tasks that are well-defined and verifiable. Its outputs can be checked against schemas. Its actions can be audited. Its errors are detectable.

The contrast with AI-as-interpreter is stark. An AI reading unstructured corporate records to extract a cap table is performing a task prone to hallucination, difficult to verify, and expensive to correct when wrong. An AI generating a cap table report from structured shareholder records is performing a task that is deterministic, trivially verifiable, and essentially error-free.

Generative AI as the Lawyer's Programming Interface

A persistent objection to data-centric legal infrastructure has been accessibility: lawyers are not programmers, and asking them to write scripts or query databases seemed unrealistic. This objection has dissolved with the emergence of generative AI coding assistants.

Large language models trained on code can translate natural language instructions into working programs. What has been coined as "vibe coding" in February 2025²⁶ is increasingly delivering real production-ready results in late 2025 and is rapidly embraced by the legal profession.²⁷ A lawyer who cannot write Python can nonetheless describe what they want to accomplish: "Resign Jane Doe from all boards in the corporate group effective March 15, generate the resolutions for each jurisdiction, and queue the registry filings." The AI assistant generates the script, explains what it does, and executes it upon approval.

This capability transforms the relationship between lawyers and structured corporate data. Not all lawyers in the team need to understand the underlying data schema in detail. They describe the business outcome; the AI translates that description into operations on the corporate repository. The lawyer reviews the proposed changes, verifies the generated documents, and approves.

Consider a concrete workflow. A partner at a law firm needs to effectuate a complex reorganization: transferring ownership of a subsidiary from one holding company to another, updating director appointments, and amending intercompany agreements. In a document-centric world, this requires drafting dozens of documents, coordinating signatures, and manually updating records.

With corporation-as-code and a generative AI assistant, the workflow becomes:

1. **Describe the transaction:** The lawyer explains the reorganization in natural language, specifying the entities involved, the effective date, and the business objectives.
2. **Generate the script:** The AI produces a script that queries the corporate repository, identifies affected entities and relationships, generates the required

²⁶ Andrej Karpathy (@karpathy), X/Twitter post, February 2025, <https://x.com/karpathy/status/1886192184808149383>

²⁷ Richard Tromans, "Jamie Tso Interview: Vibe-Coding Your Own Legal AI Tools," Artificial Lawyer, January 5, 2026, <https://www.artificiallawyer.com/2026/01/05/jamie-tso-interview-vibe-coding-your-own-legal-ai-tools/>

changes, and produces the necessary documents (transfer agreements, board resolutions, amended shareholder registers).

3. Generate validation tests: Critically, the AI also generates tests to verify that the resulting corporate structure is valid. These tests check invariants: Does every subsidiary have a parent? Do ownership percentages sum to 100%? Are all required officers appointed? Do intercompany agreements reference entities that still exist in their stated relationships?

```
def test_reorganization_validity(corporate_group, transaction):
    """
    Validate the corporate structure after the proposed reorganization.
    Uses transaction.effective_date as the reference point.
    """
    errors = []
    effective = transaction.effective_date

    # Test: Every subsidiary must reference a valid parent
    for entity in corporate_group.subsidiaries:
        parent = get_entity(entity.parent_id)
        if parent is None:
            errors.append(f"{entity.name} references non-existent parent")

    # Test: Ownership percentages must sum to 100% (simplified; assumes
    # single class, no treasury shares, fully paid)
    for entity in corporate_group.all_entities:
        total_ownership = sum(s.percentage for s in entity.shareholders)
        if not is_close(total_ownership, 100.0):
            errors.append(f"{entity.name} ownership sums to {total_ownership}%")

    # Test: All directors must have valid appointments as of effective date
    for director in corporate_group.all_directors:
        for appointment in director.appointments:
            entity = get_entity(appointment.entity_id)
            if entity is None:
                errors.append(f"{director.name} appointed to non-existent entity")
            if appointment.end_date and appointment.end_date < effective:
                if director.status == "active":
                    errors.append(f"{director.name} marked active but appointment
expired")

    # Test: Intercompany agreements reference valid parties
    for agreement in corporate_group.intercompany_agreements:
        for party_id in agreement.party_ids:
            if get_entity(party_id) is None:
                errors.append(f"Agreement {agreement.id} references non-existent
party")

    return ValidationResult(errors=errors, passed=len(errors) == 0)
```

4. Review and approve: The lawyer reviews the generated script, the proposed changes, and the validation results. The proposed changes are presented as a DIFF report that precisely enumerates the intended edits to the corporate data structure, identifying affected entities, modified relationships, and value-level changes. This DIFF enables the lawyer to verify intent without reading or understanding the underlying script. The AI assistant can explain any aspect of the script in plain language on request. If the tests pass, the DIFF is correct, and the source documents

are accurate, the lawyer approves and the transaction is committed atomically to the corporate repository.

5. Generate instruments: The system renders the authoritative data changes as traditional legal instruments: signed resolutions, transfer agreements, updated share certificates. These documents are generated from the structured data, ensuring consistency. They serve as the human-readable record of actions whose authoritative representation is the underlying data.

This workflow was difficult to imagine even five years ago. Lawyers interacting with data structures through natural language, generating custom scripts for complex transactions, and producing validated corporate changes: this required either programming expertise or expensive custom software development. Generative AI coding assistants have democratized access to programmatic corporate operations.

The implications are significant. Lawyers can now work with structured corporate data without becoming programmers. They can describe transactions in the language they already use and receive working implementations. They can generate tests that verify their work, catching errors before they propagate. The barrier between legal expertise and technical implementation has become permeable.

Furthermore, these scripts become reusable templates. A complex reorganization script, once generated and validated, can be parameterized and saved as a "corporate action" or "agent" template with input variables: director name, effective date, consideration amount, target entity. The next similar transaction requires only specifying these variables rather than describing the transaction from scratch. Law firms and corporate legal departments can build libraries of validated transaction templates, each representing accumulated expertise about how to execute particular corporate actions correctly. A "director resignation across corporate group" agent, a "subsidiary formation" agent, a "series financing" agent: these become standardized, tested, reusable components that encode best practices and reduce the risk of procedural error.

Structure Before Intelligence

The argument is not that AI is bad, far from it. The argument is that structure should precede intelligence. The first step is representing corporate information in structured form. The second step is deploying AI to operate on that structure.

Inverting this order (deploying AI to interpret unstructured information) is an inefficient detour. It introduces risk where none is necessary. It consumes computational resources to recover information that could have been explicit. It places the burden of verification on humans who might otherwise trust the system.

Corporation-as-code gets the order right. Structure the data. Then deploy agents to maintain, query, and act on that data. The result is AI that enhances certainty rather than undermining it.

IX. Interoperability, Standardization, and the Role of the State

Organizations can implement corporation-as-code today using existing tools and formats. The core benefits (version control, auditability, automated document generation, machine-readable corporate history) are achievable immediately with zero dependency on external infrastructure. Government adoption of structured data APIs would enhance interoperability and reduce friction at regulatory touchpoints, but these are accelerants, not prerequisites.

Value Independent of Infrastructure

Corporation-as-code delivers immediate benefits independent of government participation. An organization maintaining its corporate state in structured formats gains instant auditability through version control, automated document generation from canonical data, machine-readable corporate history, trivial backup and migration, and a foundation for future interoperability. The value proposition stands on its own: a founder can manage their entire corporate structure in Git with markdown and JSON files, generate all required documents programmatically, and migrate to any future standard with straightforward code.

Government APIs as Accelerants

While not prerequisites, government APIs that expose structured corporate data amplify the benefits of corporation-as-code. Corporate data flows through government systems at critical points: incorporation, annual filings, regulatory disclosures, beneficial ownership reporting. When governments expose this information through APIs and accept filings in structured formats, they reduce compliance friction and enable ecosystem innovation.

The current state of government corporate services is uneven. The UK Companies House and SEC EDGAR provide robust APIs, while many U.S. state registries remain PDF-centric despite maintaining structured data internally. The United Kingdom's Companies House demonstrates what's possible through its comprehensive REST API exposing company data in JSON format.²⁸

The U.S. SEC: Government-as-API at Scale

The U.S. Securities and Exchange Commission's EDGAR system demonstrates what government corporate data infrastructure can look like. The SEC provides RESTful APIs exposing filing data in JSON format, without authentication or API keys.²⁹ Developers can query for complete filing histories, financial concepts across companies, and cross-company comparisons. Data updates in real-time (within seconds of filing dissemination). Bulk downloads of the entire dataset are available nightly. For public companies, the SEC has already built what corporation-as-code

²⁸ UK Companies House, "API Overview," <https://developer.company-information.service.gov.uk/overview/>

²⁹ SEC, "EDGAR Application Programming Interfaces," <https://www.sec.gov/search-filings/edgar-application-programming-interfaces>

envisions: the question is why state registries don't provide equivalent APIs for private company records.

Variable Progress Elsewhere

Other jurisdictions show varying degrees of progress. Corporations Canada provides XML and JSON formats for certain filings. ASIC in Australia offers API access for some registry functions. Yet many U.S. state registries remain document-first, returning PDF certificates and scanned documents rather than structured data.

International Standards Taking Hold

Cross-border standardization is already happening through market adoption. The Legal Entity Identifier (LEI), coordinated by GLEIF and endorsed by the G20 and Financial Stability Board, provides a unique 20-character code for every legal entity in financial markets, with a free, public global directory.³⁰ The Beneficial Ownership Data Standard (BODS) offers an open schema for ownership information in JSON.³¹ OASIS reports that UBL (Universal Business Language) underpins an estimated \$11 billion market for standardized business document exchange and is used in over 190 countries.³² If invoices can be standardized globally, so can corporate filings.

In practice, entity normalization will likely evolve as a blend of shared standards and local mappings. Organizations can maintain internal IDs and schemas that fit their operational needs, while exposing translation layers (or mapping tables) that support cross-system entity resolution. This pattern is common in financial services, where firms use internal identifiers but map to LEI for external reporting, and in e-commerce, where vendors rely on internal SKUs alongside standard product identifiers.³³

As organizations start exposing corporate data through APIs or via MCP-style tool integrations, market incentives may encourage convergence around a small number of widely adopted identifiers and schemas. A more practical question is which conventions reach critical mass through adoption and network effects, and where plural standards persist with adapters. Where ISO identifiers already exist (e.g., LEI, country codes), they are natural coordination points for interoperability. In domains without established ISO standards, widely implemented open schemas may function as de facto standards in practice. Organizations that invest in a coherent, well-documented internal data model lower the marginal effort required to adopt emerging external standards.

Regulatory Context: The EU Data Act

The EU Data Act (Regulation 2023/2854), effective September 2025, mandates that connected products make data available "by default, easily, securely, free of charge, in

³⁰ GLEIF, "Introducing the Legal Entity Identifier (LEI)," <https://www.gleif.org/en/organizational-identity/introducing-the-legal-entity-identifier-lei>

³¹ Open Ownership, "Beneficial Ownership Data Standard," <https://standard.openownership.org/>

³² OASIS Open, <https://www.oasis-open.org/>

³³ ISDA and GFMA, "LEI Implementation: Industry Survey for FSB Peer Review" (October 2018), https://www.isda.org/a/brvEE/ISDA_GFMA_FSB-Peer-Review_LEI-Implementation_3-October-2018_FINAL_Public.pdf; GS1 US, "Guide to UPCs," <https://www.gs1us.org/upcs-barcodes-prefixes/guide-to-upcs>

a comprehensive, structured, commonly used and machine-readable format."³⁴ The regulation mandates API-based access, 30-day switching rights between service providers, and quality equivalence for exported data. While focused on connected products, the EU Data Act establishes a regulatory precedent for codifying data portability requirements. Whether such mandates prove effective will emerge over time, but they demonstrate that structured data export obligations can be formalized in law. Similar principles could hypothetically apply to corporate data held by formation services, registered agents, and entity management platforms, though market-driven adoption may obviate such requirements.

Why Government APIs Matter

Governments serve as the authoritative source for corporate existence. When they expose structured APIs, the benefits extend in multiple directions: reduced compliance costs for businesses, automated validation for regulators, transparency for the public, and innovation platforms for the private sector. Jurisdictions that provide such infrastructure gain competitive advantages in attracting incorporations and enabling their business ecosystems.

A Pragmatic Path Forward

The ideal end state involves corporations maintaining records as structured data, with government registries accepting filings in structured formats. But this vision does not depend on universal adoption. Organizations can implement corporation-as-code today by representing their corporate state as structured data from inception.

Even without standardized APIs across jurisdictions, a well-designed schema provides immediate benefits. A founder maintaining their corporate structure in a sound data model can convert to any eventual standard with straightforward code. The value proposition stands independent of government infrastructure, though it compounds when registries provide APIs.

The examples of jurisdictions that have embraced structured data (UK Companies House, parts of Canada and Australia) demonstrate feasibility and provide models. As more organizations adopt structured approaches, network effects will drive standardization. The practical path is to start now, with existing tools and formats, building the foundation that future interoperability will require. Waiting for universal infrastructure would be waiting indefinitely.

³⁴ Regulation (EU) 2023/2854 (Data Act), <https://eur-lex.europa.eu/eli/reg/2023/2854/oj/eng>

X. Risks, Objections, and Governance Concerns

The corporation-as-code model raises legitimate concerns about evidentiary validity, security, complexity, and technosolutionism.

Legal Status

Corporation-as-code requires no legislative change. A corporation can maintain its records as structured data today under existing law. Governments do not prescribe internal record-keeping methods; they prescribe regulatory filing formats. A corporation using version-controlled repositories, structured databases, and signed commits remains compliant as long as it produces required filings in expected formats.

Common-law jurisdictions already recognize electronic records as legally equivalent to paper. The U.S. ESIGN Act, Canada's evidence acts, the EU's eIDAS regulation, and Australia's Electronic Transactions Act all accommodate electronic records.³⁵³⁶³⁷³⁸ Nothing in these frameworks mandates a particular file format such as PDF; they are generally technology-neutral and would equally accommodate JSON records that meet the applicable integrity and accessibility requirements.

As discussed earlier, Delaware's Section 224 explicitly authorizes diverse record-keeping technologies, treating paper as derivative output.³⁹⁴⁰

Security

Structured corporate data presents security concerns: databases can be breached, APIs abused, version control compromised. These risks are real but not unique. Traditional records face analogous threats: document systems can be hacked, email compromised, physical files stolen.

Mitigations are also analogous: access controls, encryption, audit logging, backups. Structured systems may offer advantages: git commits cannot be silently modified, cryptographic signatures provide tamper evidence, schema validation prevents data corruption. The key is intentional security design.

Complexity

Structured data systems may seem complex, but complexity resides in infrastructure, not user experience. Users fill out forms; systems generate appropriate representations. Well-designed tools can reduce complexity through validation rules, automated document generation, and error prevention.

Legal professionals may resist unfamiliar approaches, but this transition is happening regardless. Modern legal practice increasingly requires working with electronic

³⁵ 15 U.S.C. § 7001 et seq.; Uniform Electronic Transactions Act (1999).

³⁶ Canada Evidence Act, R.S.C. 1985, c. C-5, ss. 31.1-31.8.

³⁷ Regulation (EU) No 910/2014 (eIDAS).

³⁸ Electronic Transactions Act 1999 (Cth).

³⁹ Del. Code Ann. tit. 8, § 224. <https://delcode.delaware.gov/title8/co01/sc07/index.html>

⁴⁰ Land & Welch, "Delaware Blockchain Corporate Records Amendments," Harvard Law School Forum on Corporate Governance (May 1, 2017).

discovery, contract management, and data analysis tools. Corporation-as-code is part of this broader evolution.

Avoiding Technosolutionism

Technology is not a panacea. Structured data cannot resolve fiduciary disputes; version control cannot substitute for judgment. Corporation-as-code addresses information infrastructure, not governance substance. Hard questions of corporate law remain hard, requiring legal judgment and institutional design no data model can replace. The claim is simply that these questions are better addressed when the factual substrate is clear.

XI. Conclusion: Toward Computable Corporate Law

The corporation is an abstraction. It exists because legal systems say it does, because courts will enforce its contracts, because regulators will accept its filings, because counterparties will trust its representations. The physical artifacts associated with a corporation (the paper certificates, the bound minute books, the folders of PDFs) are not the corporation itself. They are records of the corporation, evidence of its existence and actions, interfaces through which humans interact with the underlying legal construct.

For most of corporate history, these artifacts were necessarily physical. Paper was the technology available for durable, portable, verifiable records. The legal system evolved to treat paper documents as authoritative because no better alternative existed.

That constraint no longer applies. We have had the tools to represent complex structures as data for decades. Databases, schemas, version control systems, APIs: these are mature technologies, proven at scale in domains far more complex than corporate recordkeeping. Corporations already are structured data, poorly encoded in document formats that obscure rather than reveal their nature.

The Argument Restated

This paper has argued for a shift in mental models.

The corporation should be understood not as a collection of documents but as an evolving data structure: a set of attributes, relationships, rules, and a history of state changes. Documents should be recognized as views over this structure, generated for human consumption but not authoritative in themselves. The authoritative representation should be the structured data, maintained with the same rigor that software engineers apply to codebases.

This shift enables a cascade of benefits. Interoperability becomes possible when corporate data conforms to standard schemas. Automation becomes practical when AI agents operate on structured records rather than interpreting unstructured documents. Compliance becomes continuous when legal architecture integrates with enterprise architecture. Corporate scalability becomes achievable when administrative overhead scales sublinearly with organizational complexity.

Generative AI has dissolved the barrier that once separated legal expertise from programmatic capability. A lawyer describing a complex multi-jurisdictional transaction in natural language can now receive working code that effectuates the changes, generates the required instruments, and validates the resulting structure. The objection that lawyers cannot work with data structures has lost its force. Lawyers need not become programmers; they need only describe what they want to accomplish. The AI translates intent into implementation, while the lawyer retains judgment over whether the implementation is correct.

The proposal is conservative in its technological claims. It does not require blockchain, artificial general intelligence, or any technology not already in widespread use. It requires only that we apply to corporate information the same disciplines we apply to other critical data: explicit schemas, version control, validation, and programmatic access.

The Path Forward

Implementation will not happen overnight. Existing corporate records are document-based. Legal professionals are trained in document-centric practice. Government registries vary in their readiness for structured data exchange. While these factors may constrain the pace at which the full benefits of the approach materialize (particularly regarding direct programmatic submission to registries), they are secondary considerations given the fundamental inevitability and internal benefits of structured corporate data.

The path forward has several concrete starting points. New corporations can be formed with structured data as the authoritative representation from day one. Born-digital companies can adopt corporation-as-code without legacy conversion. Entity management platforms can expose open data formats alongside their proprietary interfaces. Government registries can extend API capabilities already demonstrated in the UK and elsewhere.

Early adopters will face friction. Courts may require explanation of unfamiliar record formats. Counterparties may insist on traditional documents. Integrations with document-centric government systems will require translation layers. These are costs of transition, not permanent barriers.

As adoption spreads, the ecosystem will develop. Standards will emerge for corporate data schemas. Tools will mature for managing corporate repositories. Legal education will incorporate data literacy. Precedents will accumulate for the evidentiary treatment of structured records. The friction of early adoption will decrease as corporation-as-code becomes normal rather than novel.

Interdisciplinary Work

Realizing this vision requires collaboration across disciplines. Lawyers understand corporate law, governance requirements, and the evidentiary needs of legal proceedings. Software engineers understand data modeling, version control, and system design. Policymakers understand regulatory frameworks and the conditions under which legal infrastructure evolves. No single discipline has all the necessary expertise.

The conversation must span these communities. Legal technologists can serve as translators, but the substantive work requires genuine interdisciplinary engagement. What corporate law concepts must be preserved in any data model? What software engineering practices apply to legal records? What regulatory changes would enable (or obstruct) structured corporate data?

This paper is an invitation to that conversation. It offers a framework for thinking about corporations as code, but it does not claim to have resolved every implementation question. The specific schemas, the governance protocols, the standards for cross-jurisdictional interoperability: these remain to be developed through collaborative work.

Taking Action

For founders: Adopt structured corporate records from incorporation. Maintain cap tables as data, version-control governance documents, share repositories with counsel rather than PDF folders.

For law firms: Pilot structured records with venture-backed startups and funds. Build expertise in data-centric practice, develop templates generating documents

from structured inputs. This creates advantages: faster closings, improved transparency, higher throughput per lawyer.

For fund managers: Pilot structured representations for new funds alongside traditional documents to build confidence.

For government registries: Extend APIs to enable structured filing submissions, publish corporate data schemas.

For legal educators: Incorporate data literacy into corporate law curricula. Teach students to think about corporations as structured entities, not just as collections of documents. The lawyers of 2035 will need fluency in both legal analysis and data modeling.

The transition will be gradual, and early adopters will face friction. But every structural change in legal practice began with practitioners willing to try something different. The first law firms to adopt word processors faced skepticism; now the skeptics have retired. Corporation-as-code is a practice improvement available to anyone willing to start.

Closing Reflection

We began with a simple observation: the legal profession has gone paperless without fundamentally rethinking its relationship to paper. PDFs emulate pages. Documents remain the source of truth. The digitization has been superficial.

The corporation-as-code proposal takes digitization seriously. It asks what corporate law would look like if we designed it for the digital age rather than adapting paper-era concepts to electronic media. The answer, we have suggested, is a legal system in which corporations exist as structured data objects, documents are generated views, and the full power of computational tools can be brought to bear on corporate governance, compliance, and analysis.

Corporations will remain legal fictions, enforced by courts and regulators, governed by humans exercising judgment. The hard questions of corporate law will remain hard. But the factual infrastructure on which those questions depend can be dramatically improved. More precise records, better interoperability, reduced administrative burden, enhanced transparency: these are achievable benefits that require only that we update our mental models.

The corporation as document is an artifact of technological limitations that no longer exist. The corporation as code is a recognition that better representations are possible. The transition will take effort, but the direction is clear.

Bibliography

Accord Project Documentation. <https://docs.accordproject.org/docs/accordproject-concepts.html>

ACUS, "Recommendation 2024-5: Using Algorithmic Tools in Regulatory Enforcement" (2024). <https://www.federalregister.gov/documents/2024/12/30/2024-31352/adoption-of-recommendations>

Anthropic, "Introducing Claude Opus 4.5" (2025). <https://www.anthropic.com/news/claude-opus-4-5>

ASIC, "Application Programming Interfaces (APIs)." <https://asic.gov.au/online-services/intermediary-information-and-support/application-programming-interfaces-apis/>

Australian Electronic Transactions Act 1999. <https://www.ag.gov.au/legal-system/electronic-signatures-documents-and-transactions>

Beauchamp-Tremblay, Xavier. "GitHub Workflow and Legal Drafting." Slaw, April 21, 2017. <https://www.slaw.ca/2017/04/21/github-workflow-and-legal-drafting/>

Bloomberg Law, "Structured Data: A Litigation Game Changer." <https://www.bloomberglaw.com/external/document/X1PD0P3S000000>

Canada Evidence Act, R.S.C. 1985, c. C-5, s. 31.1. <https://laws.justice.gc.ca/eng/acts/c-5/section-31.1.html>

CGI, "Canonical Data Model Whitepaper" (2021). <https://www.cgi.com/sites/default/files/2021-12/canonical-data-model-whitepaper-2021-en.pdf>

Chen et al., "Evaluating Large Language Models Trained on Code" (2021). <https://arxiv.org/abs/2107.03374>

Columbia Business Law Review, "DAO Liability." <https://journals.library.columbia.edu/index.php/CBLR/announcement/view/564>

CommonAccord.org. <https://www.commonaccord.org/>

Corporations Canada, "Accessing Federal Corporation JSON Datasets." <https://ised-isde.canada.ca/site/corporations-canada/en/accessing-federal-corporation-json-datasets>

Delaware General Corporation Law § 224. <https://delcode.delaware.gov/title8/coo1/sco7/index.html>

Dutilh, Ari. "Solo Founders in 2025." Solo Founders, December 30, 2025. <https://solofounders.com/blog/solo-founders-in-2025-why-one-third-of-all-startups-are-flying-solo>

GLEIF, "Introducing the Legal Entity Identifier (LEI)." <https://www.gleif.org/en/organizational-identity/introducing-the-legal-entity-identifier-lei>

GS1 US. "Guide to UPCs." <https://www.gs1us.org/upcs-barcodes-prefixes/guide-to-upcs>

- Harvard Law School Forum on Corporate Governance, "A Primer on DAOs" (2022). <https://corpgov.law.harvard.edu/2022/09/17/a-primer-on-daos/>
- Land, Allison L. and Edward P. Welch. "Delaware Blockchain Corporate Records Amendments." Harvard Law School Forum on Corporate Governance (May 1, 2017).
- Huang et al., "A Survey on Hallucination in Large Language Models" (2023). <https://arxiv.org/abs/2311.05232>
- IBM, "Structured vs. Unstructured Data: What's the Difference?" <https://www.ibm.com/think/topics/structured-vs-unstructured-data>
- ISDA and GFMA. "LEI Implementation: Industry Survey for FSB Peer Review." October 2018. https://www.isda.org/a/brvEE/ISDA_GFMA_FSB-Peer-Review_LEI-Implementation_3-October-2018_FINAL_Public.pdf
- Journal of Corporate Finance, "DAO Governance" (2025). <https://www.sciencedirect.com/science/article/pii/S0929119925000021>
- Karpathy, Andrej (@karpathy). X/Twitter post on "vibe coding," February 2025. <https://x.com/karpathy/status/1886192184808149383>
- Mata v. Avianca, Inc., No. 22-cv-1461 (S.D.N.Y. 2023). <https://law.justia.com/cases/federal/district-courts/new-york/nysdce/1:2022cv01461/575368/54/>
- Microsoft Azure Blog, "Claude Opus 4.5 in Microsoft Foundry" (2025). <https://azure.microsoft.com/en-us/blog/introducing-claude-opus-4-5-in-microsoft-foundry/>
- Microsoft, "Common Data Model - model.json." <https://learn.microsoft.com/en-us/common-data-model/model-json>
- MIT Computational Law Report, "CodeX Computable Contracts." <https://law.mit.edu/codex-computable-contracts-and-insurance>
- Morris, Jason. "Blawx: Rules as Code Demonstration." MIT Computational Law Report, August 14, 2020. <https://law.mit.edu/pub/blawxrulesascodedemonstration/release/1>
- NIST, "Privacy Framework" (2020). <https://www.nist.gov/privacy-framework>
- OASIS Open. <https://www.oasis-open.org/>
- OECD OPSI, "Cracking the Code: Rulemaking for Humans and Machines" (2020). <https://oecd-opsi.org/publications/cracking-the-code/>
- Open Ownership, "Beneficial Ownership Data Standard." <https://standard.openownership.org/>
- Opara-Martins, Justice, Reza Sahandi, and Feng Tian. "Critical analysis of vendor lock-in and its impact on cloud computing migration: a business perspective." Journal of Cloud Computing 5, no. 1 (2016): 4. <https://doi.org/10.1186/s13677-016-0054-z>
- Open Policy Agent Documentation. <https://www.openpolicyagent.org/docs>
- OpenAI, "Hebbia's Deep Research Automates 90% of Finance and Legal Work, Powered by OpenAI." <https://openai.com/index/hebbia/>
- OpenFisca. <https://openfisca.org/en/>

Regulation (EU) No 910/2014 (eIDAS). <https://eur-lex.europa.eu/eli/reg/2014/910/oj/eng>

Regulation (EU) 2023/2854 (Data Act). <https://eur-lex.europa.eu/eli/reg/2023/2854/oj/eng>

Rothmann, Daniel. "Company as Code." 42 Futures, February 25, 2025. <https://blog.42futures.com/p/company-as-code>

ScienceDirect, "Vendor Lock-in in Cloud Computing." <https://www.sciencedirect.com/science/article/pii/S266596382400068X>

Stanford CodeX, "Computable Contracts Initiative." <https://law.stanford.edu/codex-the-stanford-center-for-legal-informatics/projects/stanford-computable-contracts-initiative/>

Stanford RegLab, "Hallucination-Free? Assessing the Reliability of Leading AI Legal Research Tools" (2024). <https://arxiv.org/abs/2405.20362>

TechCrunch, "Anthropic Releases Opus 4.5" (2025). <https://techcrunch.com/2025/11/24/anthropic-releases-opus-4-5-with-new-chrome-and-excel-integrations/>

The Open Group, "TOGAF Standard." <https://www.opengroup.org/togaf>

Tromans, Richard. "Jamie Tso Interview: Vibe-Coding Your Own Legal AI Tools." Artificial Lawyer, January 5, 2026. <https://www.artificiallawyer.com/2026/01/05/jamie-tso-interview-vibe-coding-your-own-legal-ai-tools/>

UK Companies House, "API Overview." <https://developer.company-information.service.gov.uk/overview/>

US SEC, "EDGAR APIs." <https://www.sec.gov/search-filings/edgar-application-programming-interfaces>

US SEC, "Inline XBRL." <https://www.sec.gov/data-research/structured-data/inline-xbrl>

World Commerce & Contracting. "SaaS Contracting Guide." June 2023. <https://www.worldcc.com/Portals/IACCM/SaaS%20Contracting%20Guide.pdf>

15 U.S.C. § 7001 (ESIGN Act). <https://www.law.cornell.edu/uscode/text/15/7001>