An Examination of Information Systems Development Based on the *Tractatus Logico-Philosophicus*

Yan Zhou, Zhentao Liu, Cheng Tang, Danlun Yang, Hengxi Mao, Yu Yang, Chunbo Wu
AVICAS Generic Technology Co., Ltd, Yangzhou, Jiangsu, China

Abstract: To address the disconnect between practical business scenarios and digital transformation objectives and existing information systems, this paper adopts Ludwig Wittgenstein's "Tractatus Logico-Philosophicus" as a theoretical framework for in-depth analysis of information system development. First, drawing on "Image Theory," it establishes analogies between data models and logical representations of the real world, emphasizing that the structural integrity of Entity-Relationship (ER) diagrams is crucial for maintaining business fidelity. This establishes the ontological foundation for data governance. Second, the paper redefines software code implementation as "executable propositions" and interface definitions as logical connectors between functional modules, demonstrating how formal logic contracts can create clear, decoupled system architectures. Building on Wittgenstein's assertion that "the boundaries of language are the boundaries of the world," it reveals the epistemological limitations of information systems in encompassing business evolution and system objectives. Finally, aligning with Wittgenstein's later philosophical shifts, the paper advocates that information system development should avoid blind functional accumulation and data storage. Instead, it emphasizes logical coherence, using clear "connection rules" to enable system operation while embracing imperfections, transforming systems into flexible, user-friendly tools rather than fragile "perfect models."

Keywords: Digitization; Information System; Logic Philosophy; Data Model

1. Introduction: From Abstract Philosophy to Concrete Engineering Practice

In the wave of digital transformation, IT development teams are tasked with transforming complex business realities into precise, automated formalized systems. This process is commonly understood as technical integration or process implementation. In digital practices, the standard approach typically follows these steps: first establishing business systems, tools, or data middle platforms, then advancing system integration, and finally implementing data governance with unified standards, single-source data, and end-to-end traceability. The core challenges we face during this development process primarily include the following two aspects:

(1) How to establish a seamless and unambiguous connection between highly abstract business intentions and rigorous formal implementation in today's rapidly evolving landscape. This necessitates transcending purely technical domains and drawing wisdom from deeper philosophical logic.

(2) With increasingly comprehensive system functionalities and data, collaboration may not necessarily become more efficient; even after system deployment, frequent reliance on offline communication, manual reconciliation, and exceptional access remains.

These two issues highlight that the core challenge in system development lies not in whether a system exists, but in whether its programming language can reliably support the practical needs of business informatization. In other words, the essence of building an informatization system isn't about its interface or functionality, but whether it possesses a sufficiently clear, shareable, and computable structure as a formal language. Sometimes we lose our way during informatization projects, mechanically following user feedback and blindly modifying systems, which often leads to structural imbalances, chaos, or even system crashes. This requires the entire informatization team-especially product managers-to not only have a deep understanding of business operations and informatization systems, but also maintain clear thinking. In today's rapidly evolving digital landscape, they must also

abstract philosophical foundations relevant to informatization development. Interestingly, Ludwig Wittgenstein, the renowned logician, studied engineering, and Alan Turing, the father of computing, was his student. This background gives Wittgenstein's analytical methods a strong digital flavor, providing a natural point of convergence for our exploration.

The core thesis of this paper is to develop a more profound and practical philosophical framework for modern information systems, particularly in business process mapping, architecture design, data governance, and interface definition. This is achieved by drawing on Wittgenstein's insightful analysis of language, logic, and the world in his "Tractatus Logico-Philosophicus". The study transforms these abstract philosophical insights into actionable design principles and governance strategies for product managers and system developers, thereby broadening the perspectives of IT system development teams and offering a potential philosophical approach to system design.

The following will systematically explore the practical value of this philosophical thinking framework from three aspects: data modeling, software architecture and system boundary, and discuss how to construct an information system with clear logic, reliable engineering and clear understanding of its own boundary.

2. The Cornerstone of Data World: Information System as the Logical Image of the World

This chapter establishes an analogy matrix that positions Wittgenstein's image theory as the theoretical foundation for understanding data modeling and governance. The reliability of an information system fundamentally depends on having a logically precise business model and data model. This model serves as a logical representation of the real world, where its fidelity directly determines the feasibility and usability of all subsequent system applications.

2.1 The Composition of the World: Facts Rather Than Things

The "Tractatus Logico-Philosophicus" opens with its seminal assertion: "The world is the sum of facts, not the sum of things." [1] This perspective finds direct application to information systems. The fundamental data types at the system's core-such as integers or Boolean values-can be viewed as the building blocks or atomic units of the digital world. However, these isolated elements hold no inherent business significance. They only acquire operational meaning when interconnected in specific ways to form structured data records, such as a complete form or an approval process. It is precisely these complex facts constructed from atomic data that systems are designed to process and model.

2.2 Entity-Relationship Model: The Concretization of Logical Images

Wittgenstein's image theory posits that an image can represent a fact only if it shares the same logical schema with that fact.[1] In information system design, Entity-Relationship (ER) diagrams serve as concrete tools for this logical framework.[2] The engineering mandate for standardized data models directly implements Wittgenstein's image theory, where the structural integrity of ER diagrams-through their logical schema-serves as the sole guarantee of fidelity to business reality. The three core elements of ER diagrams-entities, attributes, and relationships-precisely mirror the logical structure of real-world business facts:

(1) Entity: represents distinguishable objects in the real world.

(2) Attributes: Describe the characteristics of an entity.

(3) Relation: It defines the association between entities, constituting the factual content itself.

An erroneous ER design, such as incorrect relational cardinality settings, is equivalent to constructing a distorted image that deviates from reality. Philosophically, this implies the proposition loses its meaning, while in engineering practice, it inevitably leads to data redundancy, inconsistency, and integrity errors, with repair costs being extremely high.

2.3 Data Governance: Maintaining Image Fidelity

If the ER model represents the logical blueprint of a system, then data governance serves as the core mechanism to maintain its clarity and fidelity. Establishing data access standards, storage protocols, and classification frameworks essentially defines the syntactic rules of this system's formal language. Wittgenstein's principle that "what can be said must be said clearly" remains central to data governance. Its essence lies in ensuring the system's language remains precise, eliminating logical confusion caused by ambiguous definitions or unclear

relationships, and preventing genuine philosophical issues (business challenges in IT infrastructure) from being obscured by so-called "grammatical illusions." By enforcing rigorous data standards, we guarantee that the logical framework of information systems consistently and unambiguously conveys business realities.

The table below summarizes the structural mapping relationship between the core concepts of "Tractatus Logico-Philosophicus" and the components of information systems [1]:

Table 1. Structural Mapping Relationship between "Tractatus Logico-Philosophicus" and Information Systems

The Core Concept of the Treatise of Logic and Philosophy	Corresponding Elements and Significance of Information System Engineering
world	Target domain/business scope: The system attempts to describe and automate a real or expected environment.
sum of facts	Data set/data lake: All describable, queryable, and storable status information within the system.
atomic fact	Minimum data unit/scalar type: a basic information unit that cannot be further decomposed (e.g., Boolean, integer).
picture	Entity relationship diagram/data model: a shared logical representation of reality.
logic model	System architecture/Interface definition: The internal rules and structures that govern how facts and propositions are connected.
The Unutterable	The ethics, purpose, or incompleteness outside the system boundary: the domain that the system cannot express or prove in its internal logic.

After establishing the static data image of the world by comparing Wittgenstein's image theory and information system, the next chapter will explore how information system constructs and executes propositions about these facts through dynamic software architecture.

3. Executable Logic: Propositional Structure of Software Architecture and Toolchain

If the data model represents the static image of the world, then the software architecture, functionalities, and code constitute the dynamic, executable propositions about this world. This chapter aims to analyze the logical essence of software architecture and the formal foundation of component interoperability, describing how code serves as a logical construct to build the intended world within a digital environment.

3.1 Software Code: Executable Proposition of Information System

Software code is a special type of executable proposition that generates outcomes through its numerical structure. This process embodies Wittgenstein's key insight: the meaning of a proposition exists independently of its truth value. [1] In software, the logical structure of code (its syntax and design) determines its meaning-specifically, the effects it aims to compute or achieve. The execution results-whether successful or expected-represent the truth value issue, which may be influenced by runtime errors, resource limitations, or logical flaws. Therefore, system architects establish logical patterns for program propositions through clear architectural blueprints before coding. A logically coherent architecture serves as the fundamental prerequisite for ensuring software maintainability and comprehensibility.

3.2 Interface Definition: Logical Connectors Between Components

Information systems are highly dependent on functional modules composed of multiple components. This mirrors how logical propositions (individual components) are connected through logical operators (e.g., AND, OR) to form a composite proposition (the entire system). In functional modules, interfaces between components serve as logical connectors, with interface definitions being the core mechanism for establishing such connections. Interface protocols provide formalized syntactic contracts [1] that precisely define components' inputs, outputs, data types, and operations [1]. They strictly confine component interactions within formal logic, enabling rigorous representation of logical structures while treating internal implementation details as logically independent elements. This mechanism ensures modularization and loose coupling of functional modules, as the validity of component connections depends solely on abstract interface protocols rather than specific implementation languages or platforms.

Although we are committed to building a logically coherent system, we must recognize that any complex formal system inherently

possesses insurmountable limitations. The next chapter will delve into these logical boundaries.

4. The Inherent Limitations of Information System: A Warning from Wittgenstein

A mature architectural design must not only focus on construction methods but also deeply comprehend the inherent limitations of its components. This chapter will integrate Wittgenstein's theory of linguistic boundaries to reveal the epistemological boundaries that all complex information systems cannot escape. Acknowledging these boundaries represents a crucial step in transitioning from idealistic fantasies to engineering realism.

4.1 The Boundaries of Language Are the Boundaries of the World

Wittgenstein's seminal assertion that "the limits of my language are the limits of my world" directly reveals the epistemological boundaries of system modeling. Every business architecture blueprint of an information system delineates a linguistic boundary, defining the entire scope of business operations that can be formally represented, stored, and processed [1]. Any business facts not captured by ER diagrams or defined by interfaces hold no logical significance for the system. However, real-world business dynamics are constantly evolving, and these unanticipated facts ultimately manifest as system failures, integration issues, or data blind spots. The consequence is that while system functionalities continue to expand, they may become unusable or even entirely unused by users. This is precisely the root cause of the inevitable linguistic conflicts between systems and the real world.

4.2 The Unspoken of the System: Ethics and Purposes

The "Tractatus Logico-Philosophicus" categorizes ethical and esthetic domains as inexpressible. Logical models can only describe what exists, but cannot articulate why [1]. Similarly, the purpose of an information system-whether to enhance user efficiency or achieve commercial value-cannot be captured by ER diagrams or any form of logic. A logically perfect and rigorously designed system may still be ethically disastrous, as its formal structure (how it works) is logically disconnected from its purpose (why it exists). Formal analysis must be guided and constrained by external value systems.

The following table summarizes the philosophical implications of the limitations of formal systems for system construction:

Table 2. Philosophical Implications of Formal System Limitations on System Construction

Philosophical Theorem/Concept	corresponding phenomena of system construction	Engineering Management and Design Enlightenment
The Language Limitation of the Treatise on Logic and Philosophy	Legacy issues/blind spots in modeling	The system design must clearly admit its unspeakable part, i.e. the business rules or background that are not formalized.
On the Logic of the Logic Philosophy	Self-reference/recursive call and deadlock	The complex system synchronization mechanism may lead to the paradox that cannot be solved at the same logical level.
Language is Image (Wittgenstein's Early Period)	Strict interface and data governance	Emphasis on clear and unambiguous grammar rules is the basis of system interoperability.
Language as a Tool (Wittgenstein's Late Period)	Practicality and Evolution of Architecture	The system should not be regarded as a static image of an ideal model, but as a dynamic set of functions serving a specific function.

After recognizing these profound limitations, we must shift from pursuing logical perfection of the ideal to a more pragmatic and resilient product design to cope with the complexity of the real world.

5. Conclusion: Product Design and Principles from Ideal Image to Practical Tool

This chapter aims to converge the previous philosophical analysis into a set of clear and operable product design, architecture principles and governance strategies for enterprise informatization construction, which will guide us to build a truly solid, reliable and valuable information system.

5.1 The Shift from Idealism to Pragmatism in the System View

Wittgenstein's philosophical thought underwent a significant transformation. His early ideas viewed language as a static representation of the world, pursuing logical purity and perfect

correspondence. However, his later thoughts regarded language as a set of tools used for specific purposes, where meaning lies in their practical application [3]. This shift advocates a pragmatic framework: a logically perfect system may be rigid and fragile, struggling to adapt to the continuous changes in real-world operations. We should view information systems as flexible toolkits whose value lies in their functionality. Therefore, the construction of information systems should not focus on the logical purity of static representations, but rather on the flexibility, efficiency, and resilience of these tools.

Based on the above analysis, we have distilled the following core principles for architectural design and system governance strategies.

5.1.1 Core Architecture Design Principles

(1) Principle 1: Logical isolation of interfaces strictly adheres to the interface definition paradigm, ensuring that interoperability between information system components relies solely on abstract formal logic contracts (interface protocols) rather than specific implementation details. This rigorous logical isolation maximizes component cohesion and minimizes coupling, while maintaining the system's logical transparency.

(2) Principle 2: The Inviolability of Data Atoms. A system's data model serves as the cornerstone of its logical fidelity. Its construction must be grounded in logically indivisible atomic facts. Any ambiguity in defining these atomic facts will lead to systemic logical inconsistencies. In practice, this requires rigorous data governance to ensure every data point corresponds to a clearly defined atomic state, thereby safeguarding the logical fidelity of the underlying world view.

5.1.2 System Governance Strategies

By adopting an adaptive architecture, the boundaries of language conflict management systems are dynamically evolving, as they inevitably interact with real-world linguistic challenges. We should implement flexible development and testing approaches that enable rapid iteration and evolution guided by pragmatic principles, rather than rigidly adhering to an inflexible blueprint. The ultimate goal of architecture isn't to permanently eliminate conflicts, but to establish a resilient framework capable of continuous adaptation and effective conflict management.

5.2 Summary and Outlook

A profound understanding of formal system boundaries serves as the conceptual foundation for building truly robust, reliable, and valuable enterprise information systems. These philosophical insights remind us that product managers of information systems are not merely logical architects, but also translators bridging the real and formal worlds, as well as boundary managers. Our goal is to develop tools that are logically clear yet pragmatically flexible, rather than fragile idealized visions.

Looking ahead, with the rise of artificial intelligence, a new challenge for computational philosophy and system architecture emerges: how to more effectively formalize elements like context and value to bridge the gap between pure logic and system objectives. This requires exploring a framework that transcends classical logic, offering greater descriptive power and adaptability. This is precisely why we must draw lessons from the philosophical insights of our predecessors about the world, and re-examine the rationale behind information infrastructure development.

References

[1] Ludwig Wittgenstein. Tractatus Logico-Philosophicus [M]. Translated by Han Linhe. Beijing: Commercial Press, 2013.

[2] Wang Shan, Sashi Xuan. Database Systems: An Introduction [M]. 5th ed. Beijing: Higher Education Press, 2014.

[3]Han Linhe. Wittgenstein's Philosophical Journey [M]. Kunming: Yunnan University Press, 1996.

