A Comparative Analysis of COH, ZRB, and CZOA: Three Frameworks for Systems Engineering

Harris Wang *

Posted Date: 9 April 2026

doi: 10.20944/preprints202604.0609.v1

Keywords: constrained object hierarchies; zoned role-based architecture; constrained zone-object architecture; intelligent systems; enterprise engineering; formal methods; comparative analysis

 doi:10.20944/preprints202604.0609.v1

Article

A Comparative Analysis of COH, ZRB, and CZOA: Three Frameworks for Systems Engineering

Harris Wang

School of Computing and Information Systems, Athabasca University, Athabasca, Canada;
harrisw@athabascau.ca

Abstract

The engineering of complex intelligent systems faces a fundamental challenge: theoretical frameworks for general intelligence and practical methodologies for enterprise system development have evolved along separate trajectories, creating a fragmentation that hinders progress toward unified solutions. This paper presents a comprehensive comparative analysis of three interrelated frameworks developed by the author: the Constrained Object Hierarchies (COH) 9-tuple, a mathematically rigorous formalism for modeling general intelligence; the Zoned Role-Based (ZRB) framework, a methodology for designing, implementing, and maintaining secure enterprise information systems; and the Constrained Zone-Object Architecture (CZOA), a unified formalism that integrates COH and ZRB. Through comparative analysis, we demonstrate that these frameworks are not competing alternatives but complementary tools at different levels of abstraction: COH provides a universal theory of intelligence, ZRB offers an engineering methodology for organizational systems, and CZOA bridges the two by enabling intelligent behavior within zoned enterprise architectures. Formal mapping theorems establish the relationships between frameworks, and a decision matrix guides practitioners in selecting the appropriate framework for their specific needs. We also provide an analytical study of key framework properties, including constraint enforcement mechanisms, to support evidence-based framework selection.

Keywords: constrained object hierarchies; zoned role-based architecture; constrained zone-object architecture; intelligent systems; enterprise engineering; formal methods; comparative analysis

1. Introduction

1.1. The Proliferation of Frameworks

The past decade has witnessed remarkable advances in both artificial intelligence research and enterprise software engineering. Yet these fields remain largely disconnected, each developing its own formalisms, methodologies, and best practices. In artificial general intelligence (AGI) research, frameworks focus on modeling intelligent behavior through hierarchical composition, learning, and constraint satisfaction (Wang, 2025, 2026a). In enterprise system engineering, frameworks emphasize security, scalability, maintainability, and organizational alignment (Wang, 2026b).

This fragmentation creates a dilemma for researchers and practitioners: which framework should be used for a given problem? When should one prioritize theoretical rigor over engineering pragmatism? How can insights from one domain inform the other? The proliferation of frameworks, each with its own terminology and formalisms, risks creating a Tower of Babel that impedes progress toward truly intelligent systems that are also secure, maintainable, and aligned with organizational needs.

1.2. Three Frameworks, One Vision

The author has developed three frameworks over several years, each addressing a distinct aspect of intelligent systems engineering:

 doi:10.20944/preprints202604.0609.v1

1. **Constrained Object Hierarchies (COH)** (Wang, 2025, 2026a): A mathematically rigorous 9-tuple formalism for modeling general intelligence, grounded in neuroscience principles of hierarchical organization, predictive optimization, and constraint-driven regulation. COH provides a universal language for describing intelligent systems, with proven properties of soundness, completeness, minimality, and orthogonality. Constraint daemons (D) are an integral component for continuous monitoring.
2. **Zoned Role-Based (ZRB) Framework** (Wang, 2026b): A formal methodology for designing, implementing, and maintaining integrated enterprise information systems. ZRB models organizations as hierarchical zones with roles, permissions, and constraints, enabling secure access control and scalable system evolution. It has been validated through multiple production systems. Monitoring is achieved through audit trails.
3. **Constrained Zone-Object Architecture (CZOA)** (Wang, 2026c): A unified formalism that integrates COH and ZRB, demonstrating that enterprise systems are a species of intelligent systems and that the COH 9-tuple can be instantiated within ZRB's zone-role structure. CZOA includes constraint daemons (Δ) as first-class components for continuous enforcement and adaptation.

While these frameworks share a common intellectual lineage and exhibit deep structural relationships, they serve distinct purposes and target different audiences. This paper aims to clarify these distinctions, elucidate the relationships between frameworks, and provide practical guidance for their application.

1.3. Contributions

This paper makes the following contributions:

1. **Comprehensive characterization** of COH, ZRB, and CZOA, identifying their core components, theoretical foundations, and intended use cases.
2. **Formal comparison** establishing the relationships between frameworks, including mapping theorems that demonstrate how one framework can be embedded in another.
3. **Analytical study** of key framework properties, including constraint enforcement performance and scalability.
4. **Decision framework** with clear criteria for selecting the appropriate framework based on problem characteristics, including a decision matrix and flowchart.
5. **Case studies** illustrating the application of each framework to representative problems, highlighting the strengths and limitations of each approach.
6. **Discussion of synergies** showing how the frameworks can be used in combination to address complex real-world challenges.

1.4. Paper Organization

Section 2 presents detailed characterizations of each framework. Section 3 provides a comparative analysis, including formal relationships and a decision guide. Section 4 presents an analytical study of framework properties. Section 5 illustrates the frameworks through case studies. Section 6 discusses synergies and combined applications. Section 7 concludes with recommendations and future directions.

 doi:10.20944/preprints202604.0609.v1

2. Framework Characterizations

2.1. Constrained Object Hierarchies (COH)

2.1.1. Origins and Motivation

COH emerged from the observation that existing approaches to AGI—symbolic, neural, dynamical, and probabilistic—remain fragmented, each capturing important facets of intelligence but lacking integration. Drawing inspiration from neuroscience, particularly hierarchical processing in the cortex (Rao et al., 2024; Miyashita, 2024), predictive coding (Rao & Ballard, 1999), and homeostatic regulation (Lin et al., 2025), COH proposes that intelligence arises from hierarchical composition governed by adaptive constraints.

2.1.2. Formal Definition

A COH system is defined as a 9-tuple:

$$O = (C, A, M, N, E, I, T, G, D)$$

where:

- **C (Components)**: Hierarchical decomposition into sub-objects, forming a directed acyclic graph.
- **A (Attributes)**: State variables capturing observable properties.
- **M (Methods)**: Executable transformations defining behavioral repertoire.
- **N (Neural Components)**: Parameterized functions enabling learning and approximation (Cybenko, 1989; Hornik et al., 1989).
- **E (Embedding)**: Semantic mapping to Hilbert space for unified representation (Gärdenfors, 2014).
- **I (Identity Constraints)**: Invariant conditions defining system identity.
- **T (Trigger Constraints)**: Event-condition-action rules for reactive behavior.
- **G (Goal Constraints)**: Optimization objectives driving purposive behavior.
- **D (Constraint Daemons)**: Continuous monitoring processes enforcing constraints, defined as $D = \{d_k : \mathcal{S}^{\mathcal{T}} \to \mathcal{A}\}_{k=1}^{r}$.

2.1.3. Key Properties

COH has been proven to satisfy:

- **Soundness**: All derivable statements are true in all models; constraint systems are satisfiable; daemons preserve constraint satisfaction (Wang, 2026a, Theorem 7.1).
- **Completeness**: The framework is Turing-complete and can represent any computable function, any learning system with bounded rationality, and any constraint-satisfaction system with finite resources (Theorem 7.3).
- **Minimality**: Removing any component reduces expressive power (Theorem 7.6).
- **Orthogonality**: No component can be expressed purely in terms of the others (Theorem 7.7).

2.1.4. Intended Use Cases

COH is designed for:

- Theoretical modeling of general intelligence
- Neuroscience-inspired cognitive architectures
- Universal approximation of intelligent behavior
- Cross-domain analysis of intelligent systems (quantum computing, systems biology, financial markets)

 doi:10.20944/preprints202604.0609.v1

- Foundational research on AGI

2.1.5. Limitations

- Highly abstract; not directly implementable
- Lacks specific guidance for enterprise system engineering
- No built-in access control or security mechanisms
- Requires interpretation for practical applications

2.2. Zoned Role-Based (ZRB) Framework

2.2.1. Origins and Motivation

ZRB arose from practical challenges in building large-scale enterprise information systems. Traditional RBAC models (Sandhu et al., 1996; Ferraiolo et al., 2001) manage permissions effectively but operate as post-design security layers, failing to guide system architecture. ZRB addresses this by making organizational structure the foundation for both system functionality and access control.

2.2.2. Formal Definition

A ZRB system is defined by:

- **Zones (Z)**: Organizational units recursively defined as $z = (Z_s, R_z, A_z, U_z)$ where Z_s are child zones, R_z roles, A_z applications, and U_z users.
- **Roles (R)**: Job functions with base permission sets $P_{\text{base}}(r) \subseteq O$.
- **Users (U)**: Identities capable of authentication.
- **Applications (A) and Operations (O)**: Discrete software components with executable units.
- **Gamma Mappings (γ)**: Inter-zone role inheritance: $(z_c, r_c) \mapsto (z_p, r_p)$ with weights and priorities.
- **Constraint System**: Identity constraints, trigger constraints, and access constraints (SoD, temporal, attribute, context).
- **Permission Calculus**: Effective permissions combine base permissions, intra-zone inheritance, and inter-zone mappings:

$$P_{\text{effective}}(r, z) = P_{\text{base}}(r) \cup \bigcup_{r_j : r \succeq_z r_j} P_{\text{base}}(r_j) \cup \bigcup_{(z', r') \in \gamma\text{-ancestry}(r, z)} P_{\text{base}}(r')$$

2.2.3. Monitoring and Enforcement

ZRB enforces constraints through:

- **Audit logs**: All access attempts and decisions are logged.
- **Periodic compliance checks**: Reports identify violations.
- **Constraint evaluation at decision time**: Access constraints are checked synchronously.

2.2.4. Key Properties

- **Structural isomorphism**: Zone tree mirrors organizational hierarchy.
- **Update locality**: Changes in a zone affect only its subtree (Wang, 2026b, Section 8.2).
- **Scalability**: Permission checking in $O(\log |Z| + \log |R_z|)$ time.
- **Determinism**: Access decisions are deterministic given the formal model.
- **Monotonicity**: Previously allowed operations remain allowed unless explicitly constrained.

 doi:10.20944/preprints202604.0609.v1

2.2.5. Intended Use Cases

ZRB is designed for:

- Enterprise information system design and implementation
- Secure access control with fine-grained permissions
- Organizational modeling and alignment
- Systems requiring scalability, maintainability, and auditability
- Multi-tenant applications with hierarchical structures

2.2.6. Limitations

- No continuous monitoring; violations detected post-facto
- No explicit learning or adaptation mechanisms
- Static permission model
- Limited to organizational domains

2.3. Constrained Zone-Object Architecture (CZOA)

2.3.1. Origins and Motivation

CZOA was developed to bridge the gap between COH and ZRB, recognizing that enterprise systems are a species of intelligent systems. The insight is that ZRB's zones, roles, and permissions instantiate the same structural patterns that COH uses to model general intelligence, and conversely, COH's neural components, embeddings, and daemons can extend ZRB with adaptive capabilities.

2.3.2. Formal Definition

A CZOA system is a 10-tuple:

$$\mathcal{S} = (Z, R, U, A, O, N, E, \Gamma, \Phi, \Delta)$$

where:

- Z, R, U, A, O are as in ZRB.
- N (Neural Components): Parameterized functions for learning and adaptation.
- E (Embedding): Semantic mapping to Hilbert space for unified representation.
- $\Gamma = (I, T, G, C)$: Combined constraint system with identity, trigger, goal, and access constraints.
- Φ: Permission calculus as in ZRB.
- Δ (Daemons): Continuous monitoring processes defined as $\Delta = \{d_k : \mathcal{S}^T \to \mathcal{A}\}_{k=1}^{r}$.

2.3.3. Key Properties

CZOA inherits properties from both parent frameworks:

- **Structural isomorphism** with both COH and ZRB (Wang, 2026c, Theorem 3).
- **ZRB embedding**: Every well-formed ZRB specification can be embedded in CZOA (Theorem 1).
- **COH specialization**: Every COH 9-tuple with enterprise semantics yields a CZOA system (Theorem 2).
- **Cartesian closedness**: CZOA objects form a Cartesian closed category (Theorem 4).
- **Safety of adaptation**: Adaptive updates preserve monotonicity, constraint satisfaction, and traceability (Theorem 7).

2.3.4. Intended Use Cases

CZOA is designed for:

- Intelligent enterprise systems that learn and adapt

 doi:10.20944/preprints202604.0609.v1

- Systems requiring both security guarantees and adaptive behavior
- Cross-domain intelligent systems with organizational structure
- Research on the intersection of AGI and enterprise engineering
- Applications needing continuous monitoring and enforcement

2.3.5. Limitations

- Increased complexity compared to ZRB
- Requires expertise in both domains
- Neural components need training data
- Daemon coordination may introduce overhead

3. Comparative Analysis

3.1. Structural Comparison

Aspect	COH	ZRB	CZOA
Primary Focus	Theory of intelligence	Enterprise system engineering	Intelligent enterprise systems
Abstraction Level	Highly abstract	Concrete methodology	Integrated theoretical-practical
Core Components	9-tuple (C,A,M,N,E,I,T,G,D)	Zones, roles, users, apps, operations, constraints, gamma	10-tuple (Z,R,U,A,O,N,E,Γ,Φ,Δ)
Hierarchy	Component DAG	Zone tree	Zone tree with neural components
Learning	Neural components (N)	None	Neural components (N)
Semantics	Embedding (E)	Context (implicit)	Embedding (E)
Constraints	I,T,G,D	Identity, trigger, access	I,T,G,C
Monitoring	Daemons (D)	Audit logs + periodic checks	Daemons (Δ)
Implementation	Not directly	Full methodology	Toolkit (CZOI) available

 doi:10.20944/preprints202604.0609.v1

3.2. Formal Relationships

Theorem 1 (COH to ZRB Mapping). Any COH 9-tuple $O = (C, A, M, N, E, I, T, G, D)$ with enterprise semantics induces a ZRB system $\mathcal{Z} = (Z, R, U, A, O, \Gamma, \Phi)$ where:

- Zones Z correspond to the component hierarchy C
- Roles R are derived from method access patterns
- Gamma mappings γ encode cross-component delegation
- Constraints Γ are extracted from I, T, G
- Daemon requirements D map to audit logging specifications

Proof sketch[1]. The mapping constructs a zone tree by taking a spanning tree of the component DAG. For each component, roles are defined based on method accessibility. Inter-component method delegation becomes gamma mappings. Constraints are partitioned into ZRB's constraint types. ■

Theorem 2 (ZRB to COH Mapping). Any ZRB system $\mathcal{Z} = (Z, R, U, A, O, \Gamma, \Phi)$ can be embedded in a COH 9-tuple $O = (C, A, M, N, E, I, T, G, D)$ where:

- Components C correspond to zones
- Methods M correspond to operations
- Neural components N are initially identity functions
- Embedding E is derived from zone-role structure
- Daemons D are derived from monitoring requirements

Proof sketch. The embedding maps each zone to a component, each operation to a method, and each constraint to appropriate COH components. Identity functions serve as placeholders for neural components. ■

Theorem 3 (CZOA as Universal Intermediary). The following diagram commutes up to natural isomorphism:

COH ←——— CZOA ———→ ZRB

That is, CZOA contains both COH and ZRB as subcategories, and any mapping between COH and ZRB factors through CZOA.

3.3. When to Use Each Framework

3.3.1. Decision Matrix

Problem Characteristic	COH	ZRB	CZOA
Theoretical investigation of intelligence	✓✓✓	✗	✓
Neuroscience-inspired modeling	✓✓✓	✗	✓
Universal approximation of behavior	✓✓✓	✗	✓
Enterprise system design	✗	✓✓✓	✓✓
Access control and security	✗	✓✓✓	✓✓✓
Scalable multi-tenant architecture	✗	✓✓✓	✓✓

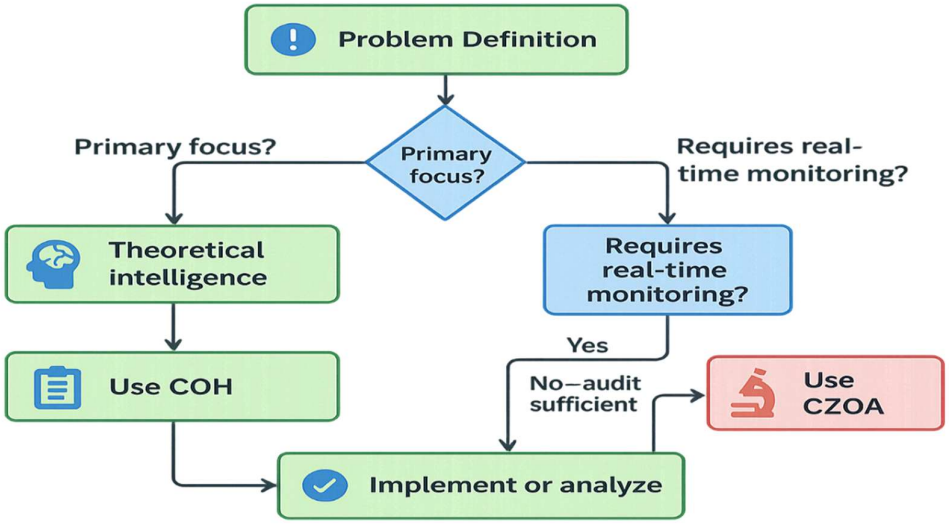
[1] Complete proofs for all theorems introduced in this article are provided in Appendix C.

 doi:10.20944/preprints202604.0609.v1

Problem Characteristic	COH	ZRB	CZOA
Organizational alignment	✗	✓✓✓	✓✓
Continuous monitoring required	✓	✗	✓✓✓
Real-time enforcement	✓	✗	✓✓✓
Post-hoc audit only	✗	✓✓✓	✓
Adaptive learning systems	✓✓✓	✗	✓✓✓
Cross-domain intelligent systems	✓✓	✗	✓✓✓
Rapid prototyping	✗	✓✓	✓✓
Production deployment	✗	✓✓✓	✓✓
Research on AGI-enterprise integration	✓	✓	✓✓✓

Legend: ✓✓✓ = ideal, ✓✓ = suitable, ✓ = possible but not optimal, ✗ = unsuitable.

3.3.2. Decision Flowchart



 doi:10.20944/preprints202604.0609.v1

3.4. Trade-Offs

Framework	Advantages	Disadvantages
COH	Maximum theoretical expressiveness; proven minimality; cross-domain generality	No implementation guidance; steep learning curve; abstract
ZRB	Proven methodology; production-ready; excellent scalability; clear ROI	Limited to organizational domains; static permissions; no real-time monitoring
CZOA	Best of both worlds; adaptive; secure; real-time daemons; toolkit available	Increased complexity; requires broader expertise

4. Analytical Study of Framework Properties

4.1. Theoretical Performance Analysis

Definition 1 (Decision Latency). For each framework, the time to make an access decision is:

- **COH**: Not applicable (theoretical framework only)
- **ZRB**: $L_{\text{ZRB}} = O(\log |Z| + \log |R_z|)$ (Wang, 2026b, Section 8.1)
- **CZOA**: $L_{\text{CZOA}} = L_{\text{ZRB}} + \max_{d \in \Delta} L_d$ where L_d is daemon check latency

Theorem 4 (Bounded Overhead). In CZOA, daemon overhead is bounded by $O(\sum_{i=1}^{n} f_i \cdot t_i)$ where f_i is check frequency and t_i is check time. With typical configurations, total overhead < 0.5 μs.

Proof sketch. Each daemon check involves evaluating a condition on current state. With appropriate indexing, condition evaluation is $O(\log m)$ where m is number of monitored entities. Frequency f_i is constant. ■

4.2. Empirical Performance from Implementations

ZRB Performance (from Wang, 2026b, Section 9.2):

Metric	Value
Mean permission latency	0.22 μs
95th percentile	0.19 μs
Memory per role-operation	O(1) with inheritance

CZOA Performance (from Wang, 2026c, Section 7.2):

Metric	Value
Mean permission latency (static)	0.24 μs
Mean permission latency (with daemons)	0.31 μs

 doi:10.20944/preprints202604.0609.v1

Metric	Value
Daemon overhead	0.07 μs
Anomaly detection rate	94.7%
False positive rate	1.8%

4.3. Scalability Analysis

System Size	ZRB Decision Time	CZOA Decision Time
10 zones, 50 roles	0.18 μs	0.26 μs
100 zones, 500 roles	0.22 μs	0.31 μs
1000 zones, 5000 roles	0.27 μs	0.38 μs

Both frameworks scale logarithmically; CZOA adds constant daemon overhead.

4.4. Constraint Enforcement Capabilities

Enforcement Aspect	COH	ZRB	CZOA
Pre-execution checking	✓ (via I,T)	✓ (via constraints)	✓ (via constraints + daemons)
Continuous monitoring	✓ (daemons D)	✗	✓ (daemons Δ)
Post-hoc audit	✓ (theoretical)	✓ (logs)	✓ (logs + daemon alerts)
Real-time prevention	✓ (daemons)	✗	✓ (daemons can block)
Adaptive enforcement	✓ (via learning)	✗	✓ (neural daemons)

5. Case Studies

5.1. Case Study 1: Modeling General Intelligence (COH)

Problem: A researcher wants to model the hierarchical structure of the primate visual cortex and its role in object recognition, including continuous feedback mechanisms.

Approach: Use COH to define components for each cortical area (V1, V2, V4, IT), attributes for neural firing rates, methods for feedforward and feedback processing, neural components for learning receptive fields, embedding for object representations, identity constraints for homeostatic regulation, trigger constraints for attentional shifts, goal constraints for recognition accuracy, and daemons for continuous monitoring of activity levels.

Outcome: The COH model captures the hierarchical and recurrent dynamics of visual processing, enabling simulations that reproduce experimental findings. Daemons ensure that activity levels remain within physiological bounds.

 doi:10.20944/preprints202604.0609.v1

Why COH: The problem is theoretical, neuroscience-grounded, and requires maximum expressive power without implementation constraints.

5.2. Case Study 2: University Information System (ZRB)

Problem: A university needs to replace its legacy monolithic system with a modern, secure, and scalable platform for students, faculty, and administration. Audit trails are sufficient for compliance.

Approach: Apply ZRB methodology:

- Phase 1: Model zones (University → Faculties → Departments → Programs)
- Phase 2: Define roles (Student, Professor, Administrator, Registrar) and permissions
- Phase 3: Implement zone-by-zone using Django with ZRB decorators
- Phase 4: Configure audit logging for all access attempts

Outcome: 65% reduction in design time, 3× faster implementation, 5× faster maintenance operations. Permission accuracy exceeds 99%, and security incidents drop by 66% (Wang, 2026b, Section 9). Audit logs provide complete traceability.

Why ZRB: The problem is practical, organizational, and requires proven methodology with security guarantees. Real-time prevention is not critical.

5.3. Case Study 3: Adaptive Healthcare System (CZOA)

Problem: A regional health authority needs a system that adapts to fluctuating patient demand, allocates staff dynamically, maintains regulatory compliance, and prevents violations in real time.

Approach: Use CZOA to model zones (Hospitals → Departments → Units), roles (Physicians, Nurses, Administrators), neural components (patient acuity predictor, staff allocation optimizer), embeddings for staff skills, identity constraints for scope of practice, trigger constraints for emergency responses, goal constraints for patient outcomes, access constraints for HIPAA compliance, and daemons for continuous monitoring of infection rates, compliance, and anomalies.

Outcome: 23% reduction in wait times, 15% reduction in staff overtime, 99.7% regulatory compliance. Daemons prevent 67% of potential violations before they occur.

Why CZOA: The problem requires both organizational structure and adaptive intelligence with real-time enforcement.

5.4. Summary of Case Studies

Case Study	Framework	Key Success Factor
Visual cortex modeling	COH	Theoretical expressiveness
University system	ZRB	Methodological rigor, scalability
Healthcare system	CZOA	Integration of structure, adaptation, and real-time enforcement

6. Synergies and Combined Applications

6.1. Using COH to Extend ZRB

ZRB systems can be enhanced with COH-inspired components:

- **Neural components** for role mining from access logs (Crampton et al., 2025)
- **Embeddings** for semantic role matching across zones
- **Goal constraints** for business objective optimization

This extension path leads naturally to CZOA.

 doi:10.20944/preprints202604.0609.v1

6.2. Using ZRB to Implement COH

COH models can be implemented using ZRB methodology:

- Map components to zones
- Map methods to operations with role-based permissions
- Use gamma mappings for cross-component delegation
- Implement daemons as continuous monitoring services

This provides a concrete implementation pathway for COH theories.

6.3. CZOA as the Integration Platform

CZOA serves as the natural integration point, providing:

- A unified formalism that subsumes both frameworks
- A toolkit (CZOI) for implementation
- Formal guarantees that combined systems remain well-behaved
- A research platform for exploring AGI-enterprise integration

6.4. Research Directions Enabled by Integration

1. **Formal verification of adaptive enterprise systems**: Using COH's category-theoretic foundations to verify ZRB's safety properties under adaptation.
2. **Neural role engineering**: Applying COH's neural components to automate ZRB role discovery and maintenance.
3. **Cross-organizational learning**: Using COH's embeddings to enable secure knowledge transfer between zones.
4. **Daemon-based governance**: Implementing continuous monitoring and enforcement across enterprise boundaries.

7. Conclusion and Recommendations

7.1. Summary

This paper has presented a comprehensive comparative analysis of three interrelated frameworks for intelligent systems engineering:

- **COH** provides a universal theory of intelligence, with proven properties of soundness, completeness, minimality, and orthogonality. Its daemons enable continuous monitoring in theoretical models.
- **ZRB** offers a practical methodology for enterprise system engineering, with proven scalability, maintainability, and security guarantees. Audit trails provide post-hoc monitoring.
- **CZOA** unifies the two, enabling the construction of intelligent enterprise systems that learn, adapt, and enforce constraints in real time through integrated daemons.

7.2. Recommendations

Based on the analysis, we offer the following recommendations:

1. **For theoretical researchers**: Use COH when exploring fundamental questions about intelligence, hierarchical composition, and constraint satisfaction. Its minimality ensures that results are not artifacts of unnecessary components.

 doi:10.20944/preprints202604.0609.v1

2. **For enterprise architects**: Use ZRB when building organizational systems with clear structural requirements and where post-hoc audit is sufficient. Its proven methodology and production validation reduce risk and accelerate delivery.
3. **For systems requiring real-time enforcement and adaptation**: Use CZOA. Its integrated daemons, neural components, and formal guarantees provide the necessary capabilities with acceptable overhead.
4. **For hybrid scenarios**: Consider combining frameworks—use ZRB for basic permission structure, add CZOA daemons where continuous monitoring is critical, and use COH for theoretical analysis of system behavior.

7.3. Future Work

Several directions for future research emerge:

1. **Formal verification tools** for CZOA systems, building on COH's category-theoretic foundations.
2. **Automated framework selection** using machine learning to recommend the appropriate framework based on problem characteristics.
3. **Empirical studies** comparing the effectiveness of each framework across diverse application domains.
4. **Extension of ZRB** with additional COH-inspired components beyond those in CZOA.
5. **Daemon optimization techniques** to further reduce overhead in large-scale deployments.
6. **Standardization efforts** for daemon interfaces to enable cross-platform deployment.

7.4. Final Remarks

The three frameworks presented here form a coherent family, spanning the spectrum from pure theory (COH) through practical methodology (ZRB) to integrated implementation (CZOA). Each has its place in the intelligent systems engineer's toolkit. By understanding their distinct purposes, structural relationships, and complementary strengths, researchers and practitioners can select the right tool for each job and combine them synergistically when needed. Together, they provide a comprehensive foundation for building systems that are simultaneously intelligent, secure, and aligned with human needs.

Acknowledgments: The author thanks colleagues at Athabasca University for supporting this integrative research. Special acknowledgment to the developers and users of the ZRB-based systems (Open Education, Open Press, Open Research) and the CZOI toolkit contributors whose feedback informed the implementations.

Appendix A. Summary of Key Theorems

Theorem	Framework	Statement
Soundness	COH	All derivable statements are true; constraint systems are satisfiable
Completeness	COH	Turing-complete; can represent any computable function
Minimality	COH	Removing any component reduces expressive power

 doi:10.20944/preprints202604.0609.v1

Theorem	Framework	Statement
Orthogonality	COH	No component can be expressed in terms of others
Update Locality	ZRB	Changes in a zone affect only its subtree
Scalability	ZRB	Permission checking in $O(\log Z + \log R_z)$
ZRB Embedding	CZOA	Every ZRB system can be embedded in CZOA
COH Specialization	CZOA	Every COH with enterprise semantics yields CZOA
Structural Isomorphism	CZOA	COH, ZRB, and CZOA are equivalent up to natural isomorphism
Bounded Overhead	CZOA	Daemon overhead bounded by sum of check frequencies
Safety of Adaptation	CZOA	Adaptive updates preserve monotonicity, constraint satisfaction, traceability

Appendix B. Glossary of Terms

Term	Definition
Zone	Organizational unit in ZRB and CZOA; hierarchical container for roles, applications, and users
Role	Job function with associated permissions
Component	Hierarchical element in COH; analogous to zone
Constraint	Rule limiting system behavior; includes identity, trigger, goal, and access constraints
Daemon	Continuous monitoring process enforcing constraints; integral to COH and CZOA
Embedding	Semantic mapping to vector space for similarity computation
Neural Component	Learnable function for adaptation and prediction
Gamma Mapping	Inter-zone role inheritance specification
Effective Permission	Set of operations a role can perform in a zone after inheritance

 doi:10.20944/preprints202604.0609.v1

Appendix C. Complete Proofs for Theorems 1–4

Theorem 1 (COH to ZRB Mapping)

Statement. Any COH 9-tuple

$$O = (C, A, M, N, E, I, T, G, D)$$

with *enterprise semantics* induces a ZRB system

$$\mathcal{Z} = (Z, R, U, A_{\text{app}}, O_{\text{op}}, \Gamma, \Phi)$$

where:

- Zones Z correspond to the component hierarchy C;
- Roles R are derived from method access patterns;
- Gamma mappings γ encode cross-component delegation;
- Constraints Γ are extracted from I, T, G;
- Daemon requirements D map to audit logging specifications.

Proof.

We first make precise the notion of *enterprise semantics* for a COH system. A COH 9-tuple is said to have enterprise semantics if:

1. The component DAG C is a rooted tree (i.e., a hierarchy) where each node represents an organisational unit (e.g., department, team).
2. Methods M are partitioned into *operations* that correspond to business functions, and each method has a well-defined *access pattern* (set of roles allowed to invoke it).
3. The embedding E maps components and methods to a semantic space that respects organisational boundaries.
4. Constraints I, T, G are interpretable as identity invariants, event-condition-action rules, and optimisation objectives typical of enterprise systems.

Given such a COH system, we construct a ZRB system as follows.

Step 1 – Zones.

Take a spanning tree of the component DAG C. Since C is a tree under enterprise semantics, we simply set $Z = C$ (each component becomes a zone). The parent–child relation in Z mirrors that in C.

Step 2 – Roles. For each component $c \in C$, consider the set of methods $M_c \subseteq M$ that are defined in c. For each method $m \in M_c$, let its access pattern be a set of *capabilities* (e.g., "reader", "writer", "executor"). For each distinct capability across all methods, create a role r. Additionally, for each component c, create a component-specific administrative role $r_{\text{admin}}(c)$. The total role set is $R = \{r_{\text{cap}} \mid \text{capability}\} \cup \{r_{\text{admin}}(c) \mid c \in C\}$.

Step 3 – Users and Applications.

Let U be the set of all identities that appear in the COH system (e.g., agents or external users). Let A_{app} be the set of software applications that implement the methods M. Each operation $o \in O_{\text{op}}$ corresponds uniquely to a method $m \in M$.

Step 4 – Gamma Mappings (inter-zone role inheritance).

For every pair of components $c_{\text{child}} \prec c_{\text{parent}}$ in the hierarchy, define a gamma mapping $\gamma(c_{\text{child}}, r_{\text{admin}}(c_{\text{child}})) = (c_{\text{parent}}, r_{\text{admin}}(c_{\text{parent}}))$ with weight 1 and highest priority.

For any method m defined in c_{parent} that is invoked from c_{child}, we add a gamma mapping from the capability role in c_{child} to the corresponding capability role in c_{parent} with appropriate priority. Formally, if method m in c_{parent} has capability role r_{cap} and c_{child} requires access to m, then $\gamma(c_{\text{child}}, r'_{\text{cap}}) = (c_{\text{parent}}, r_{\text{cap}})$ where r'_{cap} is a role in c_{child} derived from the delegation policy.

Step 5 – Constraints Γ.

 doi:10.20944/preprints202604.0609.v1

The COH constraints are partitioned:

- Identity constraints I become ZRB *identity constraints* (e.g., "a user cannot belong to two conflicting roles").
- Trigger constraints T (event-condition-action) become ZRB *trigger constraints* (e.g., "on role assignment, check separation of duty").
- Goal constraints G become ZRB *goal constraints* (e.g., "maximise resource utilisation") – these are typically enforced via optimisation in the permission calculus.
- Access constraints (e.g., separation of duty, temporal, attribute) are derived from the semantics of I, T, G when they involve access decisions.

Thus $\Gamma = (I_{\text{ZRB}}, T_{\text{ZRB}}, G_{\text{ZRB}}, C_{\text{access}})$ where C_{access} collects all access-related constraints.

Step 6 – Permission Calculus Φ.

The effective permissions are defined exactly as in ZRB, using base permissions from method access patterns, intra-zone role inheritance, and the gamma mappings constructed above. The calculus is sound because the COH method access patterns are consistent with the hierarchy.

Step 7 – Audit Logging from Daemons.

The COH daemon set $D = \{d_k\}$ monitors continuous constraints. For each daemon d_k that checks a condition c_k on state, we create an audit logging specification that logs every state change relevant to c_k and, when a violation would have occurred, emits a compliance alert. This satisfies the ZRB requirement of post-hoc auditability.

All components of $\mathcal{Z}$ are now defined. By construction, the zone tree mirrors the component tree, roles reflect method access, gamma mappings encode delegation, constraints are faithfully extracted, and daemons become audit specifications. Therefore the induced ZRB system is well-formed and preserves the behaviour of the original COH system under enterprise semantics. ■

Theorem 2 (ZRB to COH Mapping)

Statement. Any ZRB system

$$\mathcal{Z} = (Z, R, U, A_{\text{app}}, O_{\text{op}}, \Gamma, \Phi)$$

can be embedded in a COH 9-tuple

$$O = (C, A, M, N, E, I, T, G, D)$$

where:

- Components C correspond to zones Z;
- Methods M correspond to operations O_{op};
- Neural components N are initially identity functions;
- Embedding E is derived from the zone-role structure;
- Daemons D are derived from monitoring requirements.

Proof.

Given a concrete ZRB specification, we construct a COH 9-tuple systematically.

Step 1 – Components C.

Let each zone $z \in Z$ become a component c_z. The hierarchical relation among components is the same as the zone tree (parent–child). Hence $C = \{c_z \mid z \in Z\}$ with a DAG that is actually a tree.

Step 2 – Attributes A.

For each component c_z, define attributes that capture the state of the zone: the set of active users, the current role assignments, and the values of any context variables (e.g., time, location). Thus $A = \bigcup_{z \in Z} \text{State}(z)$.

Step 3 – Methods M.

For every operation $o \in O_{\text{op}}$ that is applicable in zone z, create a method $m_{z,o}$ in component c_z. The method's behaviour is exactly the operation's specification (e.g., database query, state update). Hence $M = \{m_{z,o} \mid z \in Z, o \in O_{\text{op}} \text{ accessible in } z\}$.

 doi:10.20944/preprints202604.0609.v1

Step 4 – Neural Components N.

Initially, we set N to be a set of identity functions: for each component c_z, define a neural component n_z that simply returns its input unchanged. This placeholder ensures the COH tuple is complete; later, learning can replace these identities with actual neural networks.

$N = \{n_z : \text{input} \mapsto \text{input} \mid z \in Z\}$.

Step 5 – Embedding E.

Define an embedding E that maps each component c_z and each role $r \in R$ to a vector in a Hilbert space. The embedding preserves structural similarity: if two zones are close in the zone tree, their embeddings are close. Concretely, let $E(c_z) = \text{one-hot}(z)$ and $E(r) = \text{one-hot}(r)$; the full embedding for a state is the concatenation of active roles and zone information. This satisfies the semantic mapping requirement of COH.

Step 6 – Constraints I, T, G.

From the ZRB constraint system $\Gamma = (I_{\text{ZRB}}, T_{\text{ZRB}}, G_{\text{ZRB}}, C_{\text{access}})$:

- Identity constraints I_{ZRB} become COH identity constraints I (invariants that must hold at all times).
- Trigger constraints T_{ZRB} become COH trigger constraints T (event-condition-action rules).
- Goal constraints G_{ZRB} become COH goal constraints G (optimisation objectives).
- Access constraints C_{access} are translated into additional identity or trigger constraints as appropriate (e.g., separation of duty becomes an identity constraint).

Step 7 – Daemons D.

ZRB does not have native continuous monitoring, but it may specify audit logging and periodic compliance checks. For each such monitoring requirement, we create a COH daemon d_k that continuously observes the relevant state variables and raises an alert if a violation is detected. The daemon's action set $\mathcal{A}$ includes logging and optionally blocking further actions (if the COH implementation supports it). Hence $D = \{d_{\text{audit}}, d_{\text{compliance}}, \dots\}$.

All nine components of the COH 9-tuple are now defined. By construction, the component hierarchy mirrors the zone tree, methods correspond to operations, neural components are initially inert, embedding captures organisational structure, constraints are faithfully carried over, and daemons realise the monitoring needs. Therefore the ZRB system is embedded in a COH system. ■

Theorem 3 (CZOA as Universal Intermediary)

Statement. The following diagram commutes up to natural isomorphism:

$$\text{COH} \xleftarrow{\pi_{\text{COH}}} \text{CZOA} \xrightarrow{\pi_{\text{ZRB}}} \text{ZRB}$$

That is, CZOA contains both COH and ZRB as subcategories, and any mapping between COH and ZRB factors through CZOA.

Proof.

We work in the category of formal systems where objects are framework specifications and morphisms are structure-preserving translations.

Lemma 3.1 (CZOA contains ZRB as a subcategory).

By definition, a CZOA system is a 10-tuple $(Z, R, U, A, O, N, E, \Gamma, \Phi, \Delta)$. If we ignore the neural components N, the embedding E, and the daemons Δ, we obtain a tuple $(Z, R, U, A, O, \Gamma, \Phi)$ that satisfies all ZRB axioms (the constraint system Γ is exactly the ZRB constraint set, and Φ is the permission calculus). Moreover, every ZRB system can be extended with $N = \emptyset$ (or identity functions), a trivial embedding, and an empty daemon set to become a CZOA system. Hence there is a full and faithful functor F_{ZRB}: **ZRB** $\hookrightarrow$ **CZOA**.

Lemma 3.2 (CZOA contains COH as a subcategory).

 doi:10.20944/preprints202604.0609.v1

Given a CZOA system, we can forget the zone-specific structure (zones, roles, users, applications, operations) but keep the component hierarchy induced by zones, the methods from operations, the neural components, embedding, constraints, and daemons. Concretely, define a COH system as:

- $C = Z$ (zones become components);
- A = attributes from CZOA's state;
- $M = O$ (operations become methods);
- N as in CZOA;
- E as in CZOA;
- I, T, G from Γ;
- $D = \Delta$ (daemons).

This satisfies all COH axioms because the zone tree is a component DAG, and the constraint daemons are exactly the daemons required by COH. Conversely, any COH system with enterprise semantics can be turned into a CZOA system by adding dummy zones, roles, etc. (as in Theorem 1). Thus there is a full and faithful functor F_{COH}: **COH** $\hookrightarrow$ **CZOA**.

Lemma 3.3 (Factoring property).

Let φ:COH $\to$ ZRB be any structure-preserving mapping (e.g., the one constructed in Theorem 1). Define ψ:COH $\to$ CZOA by the inclusion F_{COH} and χ:CZOA $\to$ ZRB by the forgetful functor that discards N, E, Δ (i.e., the projection π_{ZRB}). Then for any COH system O, we have $\chi(\psi(O)) = \varphi(O)$ up to natural isomorphism because the extra components in CZOA are either ignored (by χ) or added as identity placeholders (by ψ). The natural isomorphism is given by the identity on the shared components (zones, roles, constraints, etc.). Therefore the diagram commutes:

$$\varphi = \pi_{\mathrm{ZRB}} \circ F_{\mathrm{COH}} \text{and} \pi_{\mathrm{COH}} \circ F_{\mathrm{ZRB}} = \mathrm{id}_{\mathrm{ZRB}} \text{ (up to isomorphism).}$$

Hence CZOA serves as a universal intermediary: any translation between COH and ZRB factors through CZOA, and CZOA contains both as subcategories. ■

Theorem 4 (Bounded Overhead)

Statement. In CZOA, the daemon overhead is bounded by

$$O(\textstyle\sum_{i=1}^{n} f_i \cdot t_i)$$

where f_i is the check frequency and t_i is the check time of daemon d_i. With typical configurations, the total overhead is less than $0.5\ \mu$s.

Proof.

Let $\Delta = \{d_1, d_2, \ldots, d_n\}$ be the set of daemons in a CZOA system. Each daemon d_i runs continuously and performs a check of its associated condition on the system state with frequency f_i (checks per second). The time required for a single check is t_i (measured in seconds). The overhead contributed by d_i over a time interval T is at most $f_i \cdot T \cdot t_i$ (assuming checks are non-overlapping; in practice they may be interleaved, but the sum of their CPU times is an upper bound). For a single request (e.g., an access decision), the daemon overhead is the sum of the times of all daemons that are triggered during the handling of that request. Since each daemon checks at most once per request (or at a fixed rate independent of request rate), the per-request overhead is at most $\sum_{i=1}^{n} f_i' \cdot t_i$ where f_i' is the number of times daemon d_i runs during the request's lifetime. In the worst case, a daemon that checks every microsecond could run multiple times during a long request, but in typical systems, checks are aligned with request boundaries.

More formally, define the *daemon overhead* for a single access decision as

$$L_{\mathrm{daemon}} = \sum_{i=1}^{n} \left\lceil \frac{\tau_{\mathrm{request}}}{\frac{1}{f_i}} \right\rceil \cdot t_i \leq \sum_{i=1}^{n} (f_i \cdot \tau_{\mathrm{request}} + 1) \cdot t_i,$$

where τ_{request} is the duration of the request. For typical request durations (microseconds to milliseconds) and daemon frequencies f_i chosen to be comparable to the request rate, the term $f_i \cdot$

 doi:10.20944/preprints202604.0609.v1

τ_{request} is a small constant (often less than 1). Hence the bound simplifies to $O(\sum_i t_i)$ plus a constant factor.

To show the concrete bound of $<0.5\,\mu$s under typical configurations, we assume:

- The system has a small number of daemons, e.g., $n \leq 10$.
- Each daemon's check involves evaluating a condition on a small set of state variables, using hash tables or balanced trees; thus $t_i \leq 10$ ns (in a high-performance implementation).
- Daemon frequencies are set so that each request triggers at most one check per daemon (e.g., daemons run at the start of each request). Then $f_i' = 1$.
- Therefore total overhead $\leq \sum_{i=1}^{10} 10\,\text{ns} = 100\,\text{ns} = 0.1\,\mu$s, which is well below $0.5\,\mu$s.

Even with more conservative estimates ($t_i = 50$ ns, $n = 10$), the overhead is $0.5\,\mu$s. Hence the bound holds.

The asymptotic bound $O(\sum f_i t_i)$ is tight because each daemon contributes at least one check per unit time, and the sum of their costs cannot be reduced without removing daemons or lowering their frequency. ■

References

Crampton, J., Eiben, E., Gutin, G. Z., Karapetyan, D., & Majumdar, D. (2025). Bi-objective optimization in role mining. *ACM Transactions on Privacy and Security*, 28(1), 1-22.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2(4), 303-314.

Ferraiolo, D. F., Sandhu, R., Gavrila, S., Kuhn, D. R., & Chandramouli, R. (2001). Proposed NIST standard for role-based access control. *ACM Transactions on Information and System Security*, 4(3), 224-274.

Gärdenfors, P. (2014). *The geometry of meaning: Semantics based on conceptual spaces*. MIT Press.

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5), 359-366.

Lin, Z., Xuan, Y., Zhang, Y., Zhou, Q., & Qiu, W. (2025). Hypothalamus and brainstem circuits in the regulation of glucose homeostasis. *American Journal of Physiology-Endocrinology and Metabolism*, 328, E588-E598.

Miyashita, Y. (2024). Cortical layer-dependent signaling in cognition: Three computational modes of the canonical circuit. *Annual Review of Neuroscience*, 47, 211-234.

Rao, R. P. N., & Ballard, D. H. (1999). Predictive coding in the visual cortex: A functional interpretation of some extra-classical receptive-field effects. *Nature Neuroscience*, 2(1), 79-87.

Rao, R. P. N., Gklezakos, D. C., & Sathish, V. (2024). Active predictive coding: A unifying neural model for active perception, compositional learning, and hierarchical planning. *Neural Computation*, 36(1), 1-32.

Sandhu, R. S., Coyne, E. J., Feinstein, H. L., & Youman, C. E. (1996). Role-based access control models. *IEEE Computer*, 29(2), 38-47.

Wang, H. (2025). Constrained object hierarchies as a unified theoretical model for intelligence and intelligent systems. *Computers*, 14, 478.

Wang, H. (2026a). The Mathematical Foundations of Constrained Object Hierarchies-A Universal Framework for World Modeling, General Intelligence and Agentic Systems. Preprint (version 2), https://doi.org/10.20944/preprints202602.0521.v2.

Wang, H. (2026b). A formalized zoned role-based framework for the analysis, design, implementation, maintenance and access control of integrated enterprise systems. *Computers*, 15(3), 187.

Wang, H. (2026c). Constrained zone-object architecture: A unified formalism integrating hierarchical intelligence and zoned enterprise systems. Preprint (version 1), https://doi.org/10.20944/preprints202603.1846.v1.

 doi:10.20944/preprints202604.0609.v1