Bounded Event-Time Execution Architecture

Mart King

SyProof Ltd

London | Edinburgh – United Kingdom

Email: King@syproof.com

ORCID: https://orcid.org/0009-0007-0197-1174

Abstract

Contemporary cyber-physical and system-of-systems environments maintain sensing, synchronisation, coordination, and distributed processing capabilities across heterogeneous operational domains. Existing representational architectures define mechanisms for observation, modelling, and state coordination; however, operational state resolution remains externalised, deferred, or distributed across independent systems. Therefore, state formation proceeds through reconciliation, eventual consistency, or retrospective reconstruction rather than through bounded event-time execution. Bounded operational state-resolution semantics are not maintained within the execution domain itself.

This paper defines a bounded event-time execution architecture and an integrated Concept of Operations (CONOPS) in which admissible actions resolve into operational system state within bounded domains. The architecture is defined at both functional and operational levels.

Internal implementation constructs, deployment-specific mechanisms, and parameterisation remain abstracted from the architectural definition. Representational Digital Twin systems constitute one operational context but do not define the architectural scope of the contribution.

The architecture defines bounded event-time execution semantics and operational state-resolution behaviour for cyber-physical and system-of-systems environments. The architecture is defined as a specification-stage operational execution architecture intended to support subsequent implementation, deployment, and empirical evaluation.

Key Words

bounded execution architecture; event-time state resolution; execution semantics; operational state transition; cyber-physical systems; system-of-systems; distributed systems; operational architectures

1 Introduction

Contemporary cyber-physical and system-of-systems environments depend on coordination and synchronisation architectures to manage operational activity across distributed domains, consistent with established systems engineering practice [1]–[5]

These environments maintain increasing levels of sensing capability, telemetry, analytical processing, and representational state coordination across physical and digital infrastructures.

Existing representational architectures define mechanisms for:

(1) observation,

(2) modelling,

(3) synchronisation,

(4) distributed coordination,

(5) representational state exchange.

Operational state resolution remains distributed across external systems, deferred reconciliation processes, or institutional mediation mechanisms [6], [7]. Therefore, state formation proceeds through eventual consistency, retrospective reconstruction, or post-process convergence in place of bounded event-time execution. Operational state-resolution semantics are not maintained within the execution domain itself.

This condition constitutes an execution gap within cyber-physical and system-of-systems environments. Existing architectures define representational coordination semantics but do not define bounded operational state-resolution semantics within the execution domain.

Event-time execution semantics are established within adjacent computational domains, including distributed-state systems and stream-processing environments [6]–[8]. These semantics define ordered processing behaviour and event-time sequencing within computational environments, but do not define bounded operational state resolution across heterogeneous operational domains.

This paper defines a bounded event-time execution architecture and associated Concept of Operations (CONOPS) for cyber-physical and system-of-systems environments. The architecture establishes bounded execution semantics through which admissible actions resolve into operational system state within the execution domain.

2 Context and Baseline

Cyber-physical systems maintain coordination architectures connecting sensing environments, distributed computational processes, and operational infrastructures [2]. Digital Twin architectures extend these capabilities through representational synchronisation and state coordination across physical and digital domains [3]–[5].

Existing representational architectures primarily maintain:

(1) observational state,

(2) synchronisation processes,

(3) analytical processing layers,

(4) distributed coordination mechanisms.

These architectures maintain operational visibility across heterogeneous environments. Operational state resolution remains external to the representational architecture itself. Consequently, state formation depends upon:

(1) deferred reconciliation,

(2) distributed validation,

(3) eventual consistency,

(4) post-process convergence [6], [7].

Distributed systems literature defines established mechanisms for consistency management and state convergence across heterogeneous computational environments [6], [7]. Stream-processing systems define ordered event-time sequencing semantics for unbounded processing environments [8]. These mechanisms govern processing behaviour and distributed state coordination but do not define bounded operational state-resolution semantics across cyber-physical execution domains.

The bounded event-time execution architecture defined in this paper addresses operational state formation within bounded execution domains rather than representational synchronisation across distributed architectures.

3 Execution Gap

Existing representational architectures define mechanisms for:

(1) observation,

(2) modelling,

(3) synchronisation,

(4) coordination,

(5) distributed state exchange.

These architectures do not define bounded execution semantics through which admissible actions resolve directly into operational system state within the execution domain.

Operational state formation consequently proceeds through:

(1) reconciliation,

(2) eventual consistency,

(3) retrospective reconstruction,

(4) external validation,

(5) post-process coordination [6], [7].

This condition separates:

representational state,

from:

operational state resolution.

As a result, the execution gap consists of the absence of bounded operational state-resolution semantics within existing representational and coordination architectures.

The execution gap is observable across cyber-physical and system-of-systems environments irrespective of implementation domain. Representational Digital Twin systems constitute one operational context in which representational synchronisation and operational state resolution remain structurally separated [3]–[5].

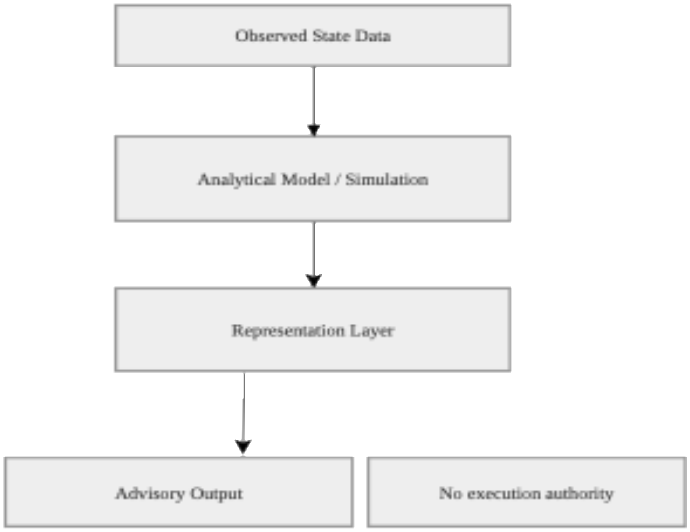
4 Bounded Event-Time Execution Architecture

The bounded event-time execution architecture defines an operational execution process through which admissible actions resolve into operational system state within bounded domains. The architecture operates as an execution layer rather than a representational architecture. It interfaces with representational systems and cyber-physical environments without depending on them for state formation.

The architecture is defined at both a functional and operational level. Internal implementation constructs, deployment-specific mechanisms, and parameterisation remain abstracted from the architectural definition. Figure 1 defines the distinction between representational open-loop architectures and bounded event-time closed-loop execution architectures.

Figure 1. Distinction between representational open-loop architectures and bounded closed-loop execution architecture.

Representational Open-Loop Architectures



Bounded Closed-Loop Execution Architecture

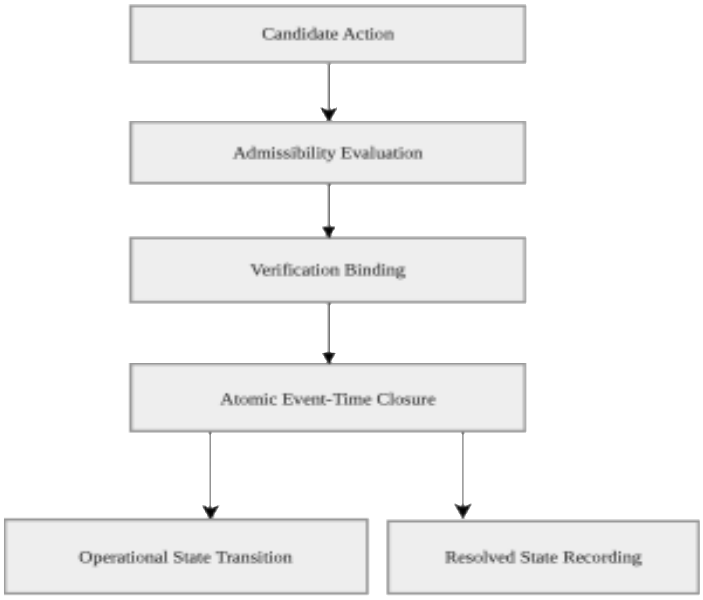


Figure 1 shows the architectural distinction between representational open-loop architectures, which maintain observation, modelling, synchronisation, and coordination across distributed environments without execution-domain state-resolution authority, and bounded closed-loop execution architectures, which define ordered execution sequences through which admissible actions resolve into operational system state within bounded domains.

Within the bounded execution architecture, candidate actions proceed through:

(1) admissibility evaluation,

(2) verification binding,

(3) event-time state resolution,

(4) operational state transition,

(5) resolved state recording.

Operational state transition occurs only upon completion of the ordered execution sequence. The architecture maintains a single resolved state within the bounded domain and does not permit post-resolution modification within the execution domain.

In this paper, “closed-loop” denotes bounded execution closure and in-domain operational state resolution, distinct from control-theoretic feedback modelling.

5 Concept of Operations (CONOPS)

The Concept of Operations defines ordered execution behaviour within the bounded domain.

5.1 Candidate Action

Execution begins with a candidate action entering the bounded execution domain.

5.2 Admissibility Evaluation

The candidate action proceeds through admissibility evaluation against operational conditions and constraints defined within the execution domain. Actions failing admissibility conditions do not proceed within the execution sequence.

5.3 Verification Binding

Admissible actions proceed through verification binding linking the candidate action to required contextual, temporal, environmental, or identity conditions within the execution domain.

5.4 Event-Time State Resolution

Verified actions proceed through event-time state resolution within the ordered execution sequence.

5.5 Operational State Transition

Operational state transition occurs only upon completion of event-time state resolution.

5.6 Resolved State Recording

Resolved operational state records are maintained within the bounded execution domain following operational state transition.

6 Discussion

Existing representational architectures maintain synchronisation and coordination across distributed cyber-physical environments [2]–[5]. Distributed systems similarly maintain mechanisms for state convergence and consistency management across heterogeneous computational environments [6], [7]. These architectures govern representation, coordination, processing, and state exchange without defining bounded operational state-resolution semantics within the execution domain itself.

The bounded event-time execution architecture defines operational state formation through ordered execution sequences maintained within bounded domains. Operational state resolves through admissibility evaluation, verification binding, event-time state resolution, operational state transition, and resolved state recording rather than through retrospective reconstruction, deferred reconciliation, or external coordination processes.

The architecture remains operationally distinct from representational coordination architectures, distributed reconciliation mechanisms, analytical synchronisation layers, and observational modelling systems. The contribution resides in the formal definition of bounded execution semantics and operational state-resolution behaviour within cyber-physical and system-of-systems environments.

The architecture remains distinct from:

(1) representational coordination architectures,

(2) distributed reconciliation mechanisms,

(3) analytical synchronisation layers,

(4) observational modelling systems.

The contribution resides in the formal definition of bounded execution semantics and operational state-resolution behaviour within cyber-physical and system-of-systems environments.

6.1 Specification Governance

The bounded event-time execution architecture is defined through a layered specification comprising:

(1) a sealed abstracted public-facing architectural specification,

(2) protected internal execution-layer constructs and kernel mechanisms, and

(3) deployment-level implementation domains.

7 Conclusion

The bounded event-time execution architecture and Concept of Operations (CONOPS) define ordered operational semantics for cyber-physical and system-of-systems environments. Existing representational architectures define coordination, synchronisation, and observational state processes across distributed domains, but do not define bounded operational state-resolution semantics within the execution domain.

The architecture establishes ordered execution semantics through which admissible actions resolve into operational system state within bounded domains. The contribution resides in the formal definition of the execution architecture and its associated operational semantics.

The architecture is presented as a specification-stage contribution supporting subsequent implementation, deployment, and empirical evaluation across heterogeneous operational environments.

Appendix A – Operational Terminology

This appendix defines operational terminology used within the bounded event-time execution architecture. Definitions maintain terminological consistency within the architectural specification and Concept of Operations (CONOPS).

Admissibility

Operational condition under which a candidate action is permitted to proceed within the bounded execution sequence.

Architectural Coordination

Coordination behaviour maintained within an operational architecture across bounded or distributed environments.

Bounded Domain

Operational environment within which execution semantics, operational constraints, and state-resolution behaviour are maintained by the execution architecture.

Closed-Loop Execution Architecture

Execution architecture maintaining operational state resolution and execution closure within the bounded domain.

Event-Time Execution

Ordered execution behaviour in which admissible actions resolve into operational system state within the execution sequence at the time of occurrence.

Event-Time State Resolution

Operational state transition occurring within the execution sequence following admissibility evaluation and verification binding.

Execution Architecture

Operational architecture governing admissibility evaluation, verification binding, event-time state resolution, operational state transition, and resolved state recording within bounded domains.

Execution Domain

Bounded operational domain within which the execution architecture maintains operational state-resolution authority.

Operational State

Resolved system condition maintained within the bounded execution domain following completion of the execution sequence.

Operational State Transition

Transition of operational system state resulting from event-time state resolution within the execution sequence.

Open-Loop Architecture

Representational architecture maintaining observation, modelling, synchronisation, or coordination without operational state-resolution authority within the execution domain.

Representational Architecture

Architecture maintaining representational state, synchronisation processes, modelling behaviour, or distributed coordination across physical and digital environments.

Resolved State Recording

Maintenance of resolved operational state records within the bounded execution domain following operational state transition.

Verification Binding

Operational verification process linking admissible actions to required contextual, temporal, environmental, or identity conditions within the execution domain.

Data Availability Statement

No empirical datasets were generated or analysed within this manuscript. The work defines a specification-stage operational execution architecture and Concept of Operations (CONOPS). Subsequent implementation and evaluation will require independent data collection.

References

[1] International Council on Systems Engineering (INCOSE),

Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities,

5th ed., Hoboken, NJ, USA: Wiley, 2023.

[2] J. Lee, B. Bagheri, and H.-A. Kao,

"A cyber-physical systems architecture for Industry 4.0-based manufacturing systems,"

Manufacturing Letters, **vol. 3, pp. 18–23, 2015.**

doi: 10.1016/j.mfglet.2014.12.001.

[3] International Organization for Standardization (ISO),

ISO 23247-1:2021 — Automation systems and integration — Digital twin framework for manufacturing — Part 1: Overview and general principles,

Geneva, Switzerland: ISO, 2021.

[4] D. Jones, C. Snider, A. Nassehi, J. Yon, and B. Hicks,

"*Characterising the digital twin: A systematic literature review*,"

CIRP Journal of Manufacturing Science and Technology, **vol. 29, pp. 36–52, 2020.**

doi: 10.1016/j.cirpj.2020.02.002.

[5] A. Fuller, Z. Fan, C. Day, and C. Barlow,

"*Digital twin: Enabling technologies, challenges and open research,"*

IEEE Access, **vol. 8, pp. 108952–108971, 2020.**

doi:10.1109/ACCESS.2020.2998358.

[6] W. Vogels, *"Eventually consistent,"*

Communications of the ACM*, **vol. 52, no. 1, pp. 40–44, 2009. doi:10.1145/1435417.1435432.

[7] M. Kleppmann,

Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems*,

Sebastopol, CA, USA: O'Reilly Media, 2017.

[8] T. Akidau, R. Bradshaw, C. Chambers, S. Chernyak, R. Fernández-Moctezuma,

R. Lax, S. McVeety, D. Mills, F. Perry, and E. Schmidt,

"*The Dataflow model: A practical approach to balancing correctness, latency, and cost in massive-scale, unbounded, out-of-order data processing*,"

Proceedings of the VLDB Endowment, **vol. 8, no. 12, pp. 1792–1803, 2015.**
doi:10.14778/2824032.2824076