



The Thread Is Not the Twin: A Doctrinal Account of Life-Cycle Information Continuity in Engineering Data Systems

Muhammad Umar Rasheed^{1*}

University of West of the Scotland, London Campus

* Email ID: umarrasheed9989@gmail.com

ARTICLE INFO

Article history:

Received: 01-08-2026

Received in revised form:
02-08-2026

Accepted: 12-09-2026

Available online: 13-09-2026

Keywords:

Asset Administration Shell
Digital Thread
Digital Twins
Engineering Data Pipelines
Life-Cycle Continuity

ABSTRACT

The term "digital twins" has become synonymous in the field of engineering informatics. However, there are many other components involved in the overall lifecycle of an asset including the quiet workhorse - the digital thread. The digital thread represents the communications framework that should provide connectivity of an asset's data throughout its lifecycle (from definition through manufacture, operation, maintenance, and eventual disposal). This paper provides a theoretical representation of this digital thread within the context of engineering informatics. The authors analyse aerospace and manufacturing thread statements against the life-cycle axis of RAMI 4.0, STEP and Asset Administration Shell practice, and enterprise-control interfaces as mutually constraining instruments. The author claims that local faithfulness of a twin can occur even when the thread is broken; however, synchronisation with a live asset does not equate to maintaining continuity among views. Three tests follow: identifier persistence, view-to-view translatability, and decision replay. The author proposes a Thread Continuity Stack as an overlay and not as a new architecture. The reconstruction of this paper was done theoretically, which enables future empirical studies to determine if a reported twin, pipeline, or operational model continues to carry a thread.

1. INTRODUCTION

Engineering organisations are fluent in speaking about digital twins. However, they tend to be less fluent when discussing the path that takes an asset's information from a type (in Design), to an instance (in Manufacturing), to Use, Maintenance, and Retirement. This path is called the digital thread. In aerospace formulations it is described as a data-driven architecture that ties together all of the information produced throughout a product's life cycle; therefore, changing how designers pose their design decision questions [1]. In an earlier U.S. Air Force statement, the digital thread was referred to as the "connective tissue" of a digital engineering enterprise - not simply another term for high-fidelity models [2]. Both terms have been repeatedly used interchangeably. The usage is convenient. However, it is an incorrect doctrinal position.

A twin, on the manufacturing classifications now widely cited, is a virtual counterpart that remains in a declared relation of update with a physical entity [3]. A thread is a continuity relation among records that were never produced at the same time, by the same role, or for the same decision. RAMI 4.0 already draws that distinction in architectural form: one axis is hierarchy, one is layers, and one is the life-cycle and value-stream passage from type to instance [4]. ISO 23247 can specify how a manufacturing twin is framed [5]. Neither instrument, read alone, tells an organisation whether last year's process plan can still explain this year's as-built inspection or next year's remaining-life estimate. That is the thread's question.

Engineering informatics now concentrates work on pipelines, twins, cyber-physical systems, intelligent operations, and informatics methods. Each of those problem-classes can succeed locally while

the thread fails. A pipeline can land this week's sensor values without remembering the type they instantiate. A twin can synchronise an instance whose design revision is no longer retrievable. An operational model can actuate on a feature store that cannot be walked back to a bill of materials. Field-level duties of provenance, fidelity, and metrological integrity have been reconstructed in neighbouring theoretical work. The present article does not repeat those reconstructions. It isolates the life-cycle duty: information must remain the same asset's information as it changes view.

Industrial practice already makes the duty expensive when it is ignored. Cloud, Internet-of-Things, and robotics stacks now promise connected operations across design, production, and service [6]. Supply-chain deployments add 5G and platform integration along the same life-cycle [7]. Connectivity is not continuity. Two plants can exchange messages and still be unable to say whether they are talking about the same instance of the same type at the same revision.

This article is based on doctrine. It makes the obligation implicit in the life-cycle architecture and the primary obligations of the thread explicit to determine how these obligations are to be carried out by engineering data systems. Four questions guide the discussion. What do the primary instruments require from a twin that it does not? Where in the normal join process is the thread broken (i.e., stripped)? How can one show that continuity exists? And how will this affect both design and future empirical manuscripts? The contributions to this paper include the Thread Continuity Stack, as well as three methods for demonstrating continuity; namely, persistent identifiers, translatability between views, and replaying decisions.

2. DOCTRINAL METHOD

Reconstructing a coherent story from primary documents as opposed to developing a new factory data set is what doctrinal research does. The primary documents used here are authoritative and nearly canonical statements of the digital thread; reference architecture documents which include a life cycle axis; and, information model practices which profess to move an asset across vendors [1].

The approach is appropriate for this theoretical construction. All that follows are claims regarding conceptual obligations not regarding measurements taken recently at a factory.

There are three categories of source material. Primary documents refer to the statements of aerospace digital engineering's digital thread; RAMI 4.0's life cycle axis; ISO 10303 (STEP), as a long running product data standard; and, the Asset Administration Shell as a digital representation spanning multiple vendors [4]. Secondary documents are twin frameworks or enterprise control models treated by other works as definitional [5]. Tertiary sources are industrial integration papers showing where life-cycle claims are made in practice [6]. Doctrine is the process of continuing to read the first category until the second cannot treat a real model as a complete thread. The method is interpretive. It does not map every 2016-2026 paper that uses the phrase. Selection favours statements that later reviews themselves treat as definitional. Claims that go beyond those instruments are labelled as proposals. Neighbouring reconstructions of provenance or metrological integrity are not reused here as if they had already settled continuity.

3. WHAT A THREAD REQUIRES THAT A TWIN DOES NOT

3.1 Two Duties, Two Timescales

Singh and Willcox treat the thread as the communication framework that allows a connected data flow and an integrated view of an asset's data across the life cycle, and they formalise the design problem that such a flow makes possible [1]. Kraft's Air Force account places the thread beside, not inside, the twin: the thread is the enterprise information coupling; the twin is a living model used for prediction and decision [2]. Kritzinger and colleagues later classified many so-called twins as models or shadows that do not even close a live loop [3]. That classification is about synchrony. The thread's problem can be present when synchrony is perfect. An instance can be updated every second from a sensor and still be cut off from the type, the process plan, and the as-designed tolerance that give the sensor meaning.

ISO 23247 organises a manufacturing twin around observable manufacturing elements, a device communication entity, a digital-twin entity, and a user entity [5]. That is the right frame for a live counterpart. It is not, by itself, a frame for asking whether the observable element is still the same individual that left the design office. RAMI 4.0's life-cycle and value-stream axis is the more precise drawing of that question [4]. Type and instance are different information objects. A thread is what makes it legitimate to treat them as stages of one asset rather than as two databases that happen to share a nickname.

3.2 Persistence, Translation, and Replay

Three constraints arise from this picture.

Firstly, IDs must remain constant. An asset that is P-441 in design, SN-00912 in manufacture, and a tag path in operations is one asset only if those names are bound. IEEE 2413's Internet of Things approach requires a common description for all domains, but does not require cleverness at the device level [8]. The Asset Administration Shell uses this common digital format to allow the asset to contain attributes, methods and documentation that are accessible to different companies [9]. A thread that cannot survive a rename is not a thread.

Secondly, views must be translatable. The geometric representations of products, manufacturing plans, inspection results and operational characteristics are different models. STEP (ISO 10303) exists because product data must be able to be transferred between applications which use different native models [10]. Translation is not a one-off extract. It is a documented relation that can be run again when the type is revised. A nightly dump that cannot be replayed after a revision is a photograph of a join, not a thread.

Third, decisions must be replayable. A life-cycle architecture is pointless if a later engineer cannot recover which information authorised a concession, a rework, or an actuation. IEC 62264 already distinguishes the enterprise-control levels at which such a decision must live [11]. Replay is not full provenance theatre. It is the ability to recover the view, the revision, and the rule that made the act legitimate. Without replay the thread can be marketed while the organisation cannot explain itself.

Table 1 states the contrast that this section defends.

Table 1. What a live twin can certify and what only a thread can certify.

Question	Twin (synchrony duty)	Thread (continuity duty)
Is the virtual entity current with this instance?	Yes, if update window and fidelity are declared [3]	Not sufficient
Is this instance still the same individual as that type and revision?	Not asked	Identifier persistence [9]
Can this week's operating feature be translated back to last year's process plan?	Not asked	View-to-view translatability [10]
Can a later engineer recover why this act was allowed?	Only if the live model still exists	Decision replay [11]

The table is the doctrinal claim. Twins remain necessary. They do not absorb the thread.

4. WHERE ENGINEERING DATA SYSTEMS BREAK THE THREAD

4.1 Type Forgotten at Acquisition

The initial split occurs at the equipment level or the history level. High speed collection is organised by tags rather than by types. This means it could have a perfect vibration signal as a time series, but there would be no connection to the gearbox type, the revision number for the housing, or the lubrication specifications that the type assumed. Therefore, smart factory systems with closed-loop multi-agents based upon large amounts of data can increase this separation (more streams, the same missing type [12]). Internet-of-Things classifications may therefore allow the integration of streams onto platforms without the need to integrate the types associated with those streams [13]. The organisation has increased its connective capabilities. However, it has decreased in ability to recognise individuals.

4.2 Instance Orphaned at Handover

The second break in the digital thread is to be a handoff from one lifecycle view of a product to another. This includes design (creating a new type) and manufacturing (creating an instance). At the service level of the lifecycle view, the product inherits a serial number and a spare parts list, which are no longer consistent with the "as built" records created during manufacture. The transition from manufacturing to service is being addressed by NIST's digital-thread oriented research in terms of manufacturing and inspection problems and not just as slogans [14].

Industry 4.0 robotics and automation narratives often assume that the handover has already been solved by connectivity [6]. Supply-chain integrations that add 5G and shared platforms make the same assumption at a longer distance [7]. Distance multiplies orphaning: a supplier's as-built is the customer's as-designed only if the identifier and the translation survive the organisational boundary. When the handoff occurs through dropping files into a contractor portal that subsequently becomes unavailable, there is an

orphaned instance. In such cases, when a duplicate of the original instance is eventually created (a 'twin') for the previously orphaned instance it may have local excellence but will still be unable to respond to inquiries made to the now deceased type.

4.3 Acts That Cannot Be Replayed

There is a third break – an operational act. Predictive maintenance, quality interventions, and supervisory control commit the process. But if the authorising feature store can't walk back to a specific version and a rule – you have a log, but you don't have a thread. It has been known for years that enterprise-control integration requires that recommendations are stored on a defined interface [11]. A dashboard that cannot be replayed once the model has been re-trained is decorative memory. Studies of the life-cycles of twin-based design, manufacture, and servicing indicate that a single connection (join) that works from one viewpoint will fail from another [15]. This failure isn't due to the fact that the twin was inaccurate; the failure is that the thread was never an object of design.

All three breaks follow a similar pattern. Informatics success (a live stream, a trained model, a current twin) is compatible with continuity failure (there is no persisting individual, there is no way to translate back into something reproducible, and there is no way to recover the operational act). Reports of the former in studies without examining the latter represent reports of successful local systems rather than life-cycle information.

5. A DOCTRINE OF THREAD CONTINUITY

5.1 Statement and Tests

The doctrine can be stated in one sentence. A digital thread exists only while an asset remains the same individual across views, those views remain translatable, and the decisions they authorised remain replayable. A twin, a pipeline, or an operational model may function when any of those conditions fail. They no longer function as a life-cycle information system.

Three tests make the sentence operational.

Identifier persistence. Can this record still be bound to one asset individual across type, instance, and operating names [9]? A pass is a maintained crosswalk that survives rename, supplier change, and serialisation. A fail is a nickname, a recycled tag, or a serial that collides across plants.

View-to-view translatability. Can a later view be computed from an earlier view by a documented, re-runnable relation after a revision [10]? A pass looks like a STEP, shell, or equivalent mapping with a dated transformation. A fail is a one-off dump, a dead portal, or a geometry that cannot be found from a feature.

Decision replay. Can a competent engineer recover the view, revision, and rule that authorised an act [11]? A pass is a record that still opens after the model is retrained. A fail is an accuracy report with no home and no history.

These tests sit beside twin-synchrony tests, not on top of them [3]. A counterpart can be current and still not belong to a thread. A stream can be well governed and still not be the same individual.

5.2 The Thread Continuity Stack

Figure 1 draws the tests as a stack along the type-to-instance passage rather than as a live-update cycle. The stack is directional because a later view cannot invent an identifier or a translation that an earlier view never wrote.

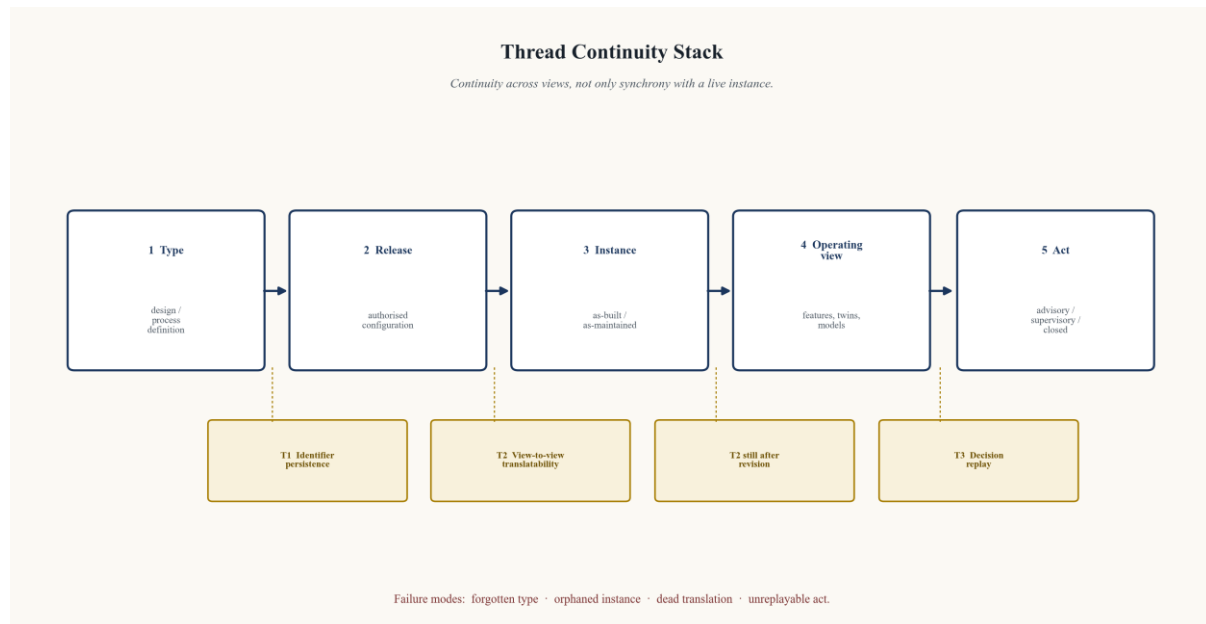


Figure 1. The Thread Continuity Stack: identifier persistence, view-to-view translatability, and decision replay must hold from type definition through instance use.

The layers are type (design and process definition), release (the authorised configuration), instance (as-built and as-maintained), operating view (features, twins, and models), and act (advisory, supervisory, or closed). Each join is policed by one or more of the three tests. The characteristic failures are a *forgotten type* at acquisition, an *orphaned instance* at handover, a *dead translation* after revision, and an *unreplayable act* at the interface.

RAMI 4.0 can host the stack on its life-cycle axis if type and instance are treated as information objects, not as labels [4]. The Asset Administration Shell can host identifier persistence if the shell is the individual, not a brochure [9]. IEC 62264 can host decision replay if the operations interface stores the view that authorised the act [11]. None of these instruments needs a successor. They need to be read as if continuity, not only connectivity, were the duty.

5.3 A Compact Vignette

A pump type is released as revision C with a bearing specification and a STEP model. Manufacturing builds serial 1844 to a mixture of revision C housings and leftover revision B bearings. The as-built file lives on a contractor share that expires with the purchase order. Two years later a plant historian streams vibration from a tag named PMP_03. A remaining-life model writes 900 hours to a dashboard. The twin of PMP_03 is current to the second.

Identifier persistence fails unless PMP_03, serial 1844, and type revision C remain bound [9]. View-to-view translatability fails when the expired share was the only translation from the STEP model to the as-built mix of C and B [10]. Decision replay fails when 900 hours cannot be recovered as a function of that mix after the model is retrained [11]. The live twin can still be excellent [5]. The thread is already broken. The vignette is schematic. It is the form of failure the three tests are built to catch.

5.4 What the Doctrine Does Not Claim

The doctrine does not require every organisation to implement a single enterprise product-lifecycle system. It requires that whatever systems are used can still answer the three tests. It does not treat STEP or the Asset Administration Shell as mandatory products. It treats them as existence proofs that translation and persistence have already been standardised [9]. It does not deny the twin. It denies that a current counterpart is a completed life-cycle record. Claims beyond that (for example a legal duty to keep every design file for the asset's physical life) are not made here.

6. DISCUSSION

Designers should assign the individual at type release, not at first sensor install [4]. Handovers should be designed as re-runnable translations with retention that outlasts the contractor [14]. Operating features

should carry the type and revision they instantiate, not only a tag path [8]. Acts should store the view that authorised them on an enterprise-control interface, not only a score [11]. Connectivity projects in robotics, cloud, and supply chain should be scored against the three tests before they are scored against message rates [6].

For research and design, the implication is narrow. Studies that claim a life-cycle, as-built, or service-coupled system should be able to say how identifier persistence, translatability, and replay were met or why they do not apply. A methodological paper that proposes a compact binding of type, instance, and operating tag for high-rate streams would be a direct sequel. A review that counts how often “digital thread” is used as a synonym for “twin” would be another [3].

The limits are those of the genre. No shipyard or factory was instrumented. The reconstruction is product-heavy because that is where type-to-instance language is most developed. Process plants, grids, and fleets will stress the same tests with different notions of individual (a batch, a feeder, a vehicle). The tests should be revised if those domains show a continuity duty the thread statements do not already name. They should not be revised merely because a platform finds expired shares convenient.

7. CONCLUSION

A digital twin may currently exist but it may still be “un-threaded”. The thread is the duty to keep an asset the same individual as its information changes view, to keep those views translatable after revision, and to keep the decisions they authorised replayable. Connectivity, live models and/or operational fidelity do not relieve this obligation - rather they can obscure that the obligation has been breached.

This article isolated the life-cycle duty for engineering informatics. It proposed three tests (identifier persistence, view-to-view translatability, and decision replay) and a Thread Continuity Stack from type to act. The ordinary failures are forgotten types, orphaned instances, dead translations, and unreplayable acts. Future empirical and methodological work can encode the stack, measure how often “thread” is used to mean “twin,” or show a domain in which one of the tests should be restated. What it should not do is treat a sensor-fed counterpart as a completed life-cycle record while the individual, the translation, and the act can no longer be recovered together.

REFERENCES

- [1] Singh, V., & Willcox, K. E. (2018). Engineering design with digital thread. *AIAA Journal*, 56(11), 4515–4528. <https://doi.org/10.2514/1.J057255>
- [2] Kraft, E. M. (2016). The air force digital thread/digital twin: Life cycle integration and use of computational and experimental knowledge. In *54th AIAA Aerospace Sciences Meeting*. American Institute of Aeronautics and Astronautics. <https://doi.org/10.2514/6.2016-0897>
- [3] Kritzinger, W., Karner, M., Traar, G., Henjes, J., & Sihn, W. (2018). Digital Twin in manufacturing: A categorical literature review and classification. *IFAC-PapersOnLine*, 51(11), 1016–1022. <https://doi.org/10.1016/j.ifacol.2018.08.474>
- [4] Deutsches Institut für Normung. (2016). *Reference Architecture Model Industrie 4.0 (RAMI4.0)* (DIN SPEC 91345:2016-04).
- [5] International Organization for Standardization. (2021). *Automation systems and integration: Digital twin framework for manufacturing: Part 1: Overview and general principles* (ISO 23247-1:2021).
- [6] Pandey, J. K., Das, S., Vats, P., Dhabliya, D., Jaiswal, S., Manikandan, K., & Pandey, D. (2024). The implications of cloud computing, IoT, and wearable robotics for smart healthcare and agriculture solutions. In N. Sindhwan, R. Anand, A. George, & D. Pandey (Eds.), *Robotics and automation in Industry 4.0* (pp. 26–45). CRC Press. <https://doi.org/10.1201/9781003317456>
- [7] Pandey, J. K., Patil, V. B., Kumari, K. A., Das, S., Namdev, A., Nandankar, P., & Gupta, A. (2023). Complete scenario for supply chain management using IoT and 5G. In P. Chawla, A. Kumar, A. Nayyar, & M. Naved (Eds.), *Blockchain, IoT, and AI technologies for supply chain management* (pp. 43–70). CRC Press. <https://doi.org/10.1201/9781003264521>
- [8] Institute of Electrical and Electronics Engineers. (2019). *IEEE standard for an architectural framework for the Internet of Things (IoT)* (IEEE Std 2413-2019). <https://doi.org/10.1109/IEEESTD.2019.8686444>
- [9] Plattform Industrie 4.0 / Industrial Digital Twin Association. (2022). *Details of the Asset Administration Shell: Part 1: The exchange of information between partners in the value chain of Industrie 4.0* (Specification). Federal Ministry for Economic Affairs and Climate Action.

- [10] International Organization for Standardization. (2014). *Industrial automation systems and integration: Product data representation and exchange: Part 1: Overview and fundamental principles* (ISO 10303-1:2014).
- [11] International Electrotechnical Commission. (2013). *Enterprise-control system integration: Part 1: Models and terminology* (IEC 62264-1:2013).
- [12] Wang, S., Wan, J., Zhang, D., Li, D., & Zhang, C. (2016). Towards smart factory for industry 4.0: A self-organized multi-agent system with big data-based feedback and coordination. *Computer Networks*, 101, 158–168. <https://doi.org/10.1016/j.comnet.2015.12.017>
- [13] Pandey, J. K., Bhuva, D., Bhuva, A., Abbas, S. H., Verma, A., & Moharekar, T. T. (2023). Classification and clustering of Internet of Things (IoT) for integrating IoT services platform. In *2023 IEEE 2nd International Conference on Industrial Electronics: Developments & Applications (ICIDEA)* (pp. 259–262). IEEE. <https://doi.org/10.1109/ICIDEA59866.2023.10295225>
- [14] Hedberg, T., Lubell, J., Fischer, L., Maggiano, L., & Feeney, A. B. (2016). Testing the digital thread in support of model-based manufacturing and inspection. *Journal of Computing and Information Science in Engineering*, 16(2), 021001. <https://doi.org/10.1115/1.4032697>
- [15] Tao, F., Cheng, J., Qi, Q., Zhang, M., Zhang, H., & Sui, F. (2018). Digital twin-driven product design, manufacturing and service with big data. *The International Journal of Advanced Manufacturing Technology*, 94(9–12), 3563–3576. <https://doi.org/10.1007/s00170-017-0233-1>