# Digital Twin Architectures: A Software Architecture Survey with Five Domain Case Studies


## Abstract

The digital twin has been described as a modelling idea, a data-integration idea, and a business proposition. It is also, and perhaps most consequentially, a software architecture problem. A digital twin is a long-lived distributed software system that must remain faithful to a physical counterpart that is itself changing, must serve several classes of stakeholder at different latencies, must integrate models written in incompatible formalisms, and must be maintained across a lifespan measured in the decades of the asset rather than the months of a release cycle. This survey reads the digital twin literature through a software architecture lens. We first isolate the conceptual commitments that constrain design: the level of integration between physical and digital counterpart, the dimensional decompositions inherited from manufacturing, and the distinction between a reference model, a reference architecture and a concrete architecture. We then map the architectural design space — the patterns actually used and their measured frequencies, the recurring layered stacks, the published pattern catalogues, the cloud/edge deployment question, and the mechanisms for composing twins into ecosystems. We treat quality attributes as the driving force, examining maintainability and evolution, latency and fidelity, interoperability, and security, and we show where these attributes conflict. Against this background we present five example architectures drawn from different domains: a manufacturing platform reference architecture, an operating-room management twin built over a legacy hospital system, a mobility twin deployed on a commercial cloud, a floating offshore wind turbine twin validated against a full-scale prototype, and a built-asset twin engineered under a DevOps regime. A cross-case comparison shows that these five systems agree on very little at the level of structure and agree substantially at the level of the problems they are solving. We close with the disagreements the literature has not resolved and the gaps the retrieved corpus does not cover.


## 1. Introduction

Software architecture is the discipline of deciding, early and deliberately, which structures a system will have, because those structures determine which qualities the system can achieve and which changes it will be able to absorb. The choice matters most when a system is long-lived, distributed, and subject to requirements from multiple stakeholders who will never be in the same room. Digital twins are exactly such systems, and the software architecture community has recognised this. A systematic mapping study of the field found a growing body of peer-reviewed architectural solutions for digital twins, screening 1630 candidate publications and analysing 140 primary studies devoted specifically to architecting them [1]. A second mapping study, taken across application domains and across software engineering for twins as a whole, kept 356 publications and classified them by contribution type, implementation method, deployment and means of evaluation [2]. That volume is itself a finding: architecting digital twins has become a research subject in its own right, distinct from modelling them or from the domain engineering that motivates them.

The reason is not hard to locate. A digital twin is not a model with a network connection. It is an assembly of heterogeneous parts that must operate as one: sensing and acquisition,

protocol adaptation, storage of both transient and persistent data, one or more executable models, an estimation or inference stage, services that consume the estimates, user interfaces, and — where the twin is genuinely a twin rather than a shadow — a return path to the physical system. Each of these parts has its own technology, its own vendor ecosystem, its own rate of change, and often its own owner. The literature describes this condition candidly. Work on human-centric industrial twins characterises the setting as one of "massive heterogeneity and fragmentation of existing systems characterized by a plethora of different devices, platforms, communication protocols and data formats" [3]. A hierarchical twin ecosystem for manufacturing frames its own motivation as the difficulty of digitalising "a complex and heterogeneous environment" given the "scattered nature of key information across the shop floor" [4].

Heterogeneity alone would be a systems integration problem. What makes it an architecture problem is that the twin has to keep integrating over time. A vision paper on engineering digital twins puts the difficulty precisely: engineering a twin "extends beyond managing the lifecycle of a Digital Twin (DT) to include its continuous, dynamic interaction with the lifecycle of the actual object, system, or process it represents", with the consequence that "when the AT changes, the DT may need to be entirely redeveloped in order not to become obsolete" [5]. Two lifecycles are coupled, and the coupling is not symmetric: the physical asset is refurbished, reconfigured, instrumented and decommissioned on a schedule that the twin's maintainers do not control. An architecture that cannot absorb changes on the physical side will be rewritten rather than evolved. The same authors identify the underlying deficiency as "the lack of a more global approach/methodology to support the engineering and evolution of DTs" [5], and dedicated notations have since been proposed to make twin evolution tractable [6].

There is a second, quieter reason to read this literature architecturally. The field has a scaling problem that is usually framed as a modelling problem but is at least as much a structural one. A widely cited perspective describes the barriers to moving digital twins "from a labor intensive and highly specialized or artisanal" product "to a widely adopted or industrial commodity", drawing its evidence from aerospace engineering and cardiology, and argues that divergent interpretation across disciplines "is limiting the wider adoption of digital-twin technologies" [7]. A complementary line of work argues that a "unifying mathematical formulation is needed to move from one-off digital twins built through custom implementations" to "robust digital twin implementations at scale", and proposes a probabilistic graphical model as that formalism [8]. Both diagnoses point at the absence of a shared abstraction. Abstraction that can be shared across instances, teams and domains is what an architecture provides.

This survey therefore asks three questions. What conceptual commitments in the digital twin literature constrain architectural choice before any structure is drawn? What is the actual design space — the patterns, layerings, deployment options and composition mechanisms that the field uses? And what do concrete architectures look like when a real system has to be built, in domains whose drivers differ as sharply as a textile shop floor differs from an operating theatre or a floating turbine?

The paper is organised accordingly. Section 2 states scope and method. Section 3 examines the concepts that constrain design. Section 4 maps the design space. Section 5 treats quality attributes and the tactics used to achieve them, including the trade-offs the literature reports. Section 6 presents five example architectures in detail. Section 7 compares them. Section 8

sets out cross-source disagreements. Section 9 gives a gap analysis, including the areas the retrieved corpus covers thinly, and Section 10 concludes.

## 2. Scope and method

This survey is drawn from a corpus of digital twin and cyber-physical systems literature retrieved by keyword-ranked search over parsed full texts. Candidate material was retrieved for eleven sub-themes — reference architectures, definitional and dimensional models, deployment styles, five application domains, standards and interoperability, quality attributes and security, composition and aggregation, model-driven engineering, platforms and tooling, evolution and lifecycle, and human-in-the-loop design — and each candidate was then read and judged individually before it was allowed to support a claim. Retrieval rank is a proxy for lexical overlap, not for relevance, and a substantial number of high-ranking hits were discarded on reading because they shared vocabulary with the query without bearing on the claim. A later whole-corpus pass ran one further search for each section of this survey, judged every candidate those searches returned in the same way, re-checked the most load-bearing quotations against their sources, and re-tested the thin-coverage judgements of Section 9 with reformulated queries.

Two boundaries should be stated. First, this is a survey of what the retrieved corpus contains, not a systematic mapping study; it does not claim exhaustive coverage of the field and it inherits whatever selection bias the underlying library carries. Where a sub-theme remained thin after reformulating queries, Section 9 says so explicitly rather than filling the space with uncited assertion. Second, the survey is deliberately architectural. It treats modelling techniques, machine learning methods and domain physics as forces acting on architecture rather than as subjects in themselves; readers wanting those subjects will find broad treatments elsewhere, including two-part reviews covering modelling and twinning enabling technologies [9] and the role of uncertainty quantification and optimisation [10].

The five example architectures in Section 6 were selected on three criteria: they come from clearly distinct domains, the source publication describes structure in enough detail to be discussed as an architecture rather than a diagram, and the source states its own limitations. The third criterion excluded several otherwise attractive candidates. It matters because an architecture whose weaknesses are undocumented cannot be compared honestly with one whose weaknesses are.

## 3. Concepts that constrain the architecture

### 3.1 Integration level is the first architectural decision

The most consequential decision in a digital twin project is made before any component is named. It is the decision about how data flows between the physical and digital counterparts, and it was given its canonical form in a categorical literature review of manufacturing twins. A *Digital Model* is “a digital representation of an existing or planned physical object that does not use any form of automated data exchange”; a *Digital Shadow* adds “an automated one-way data flow” from physical to digital; and only when “the data flows between an existing physical object and a digital object are fully integrated in both directions” is the term *Digital Twin* warranted, in which case “the digital object might also act as controlling instance of the physical object” [11].

This is a taxonomy of terminology, but it functions as a taxonomy of architecture. Each level implies a different structure. A digital model requires no runtime coupling and can be a batch artefact. A digital shadow requires acquisition, transport, storage and a synchronisation policy, and it inherits every availability and latency concern that comes with them, but it needs no return path and therefore no authority model over actuation. A digital twin requires the return path, and with it command validation, safety interlocks, arbitration between the twin and other controllers, and a security posture that treats the twin as part of the control system rather than as an observer of it. Adding bidirectionality late is not an increment; it is a redesign.

The review's most sobering result is quantitative. Of the surveyed publications, 28% describe digital models and 35% digital shadows, and "only 18 percent of them are really describing a Digital Twin with a bidirectional data-transfer" [11]. A reader of the architecture literature should hold this figure in mind, because papers routinely use the word "twin" for systems whose architecture supports only shadowing. A more recent framing proposes "twinning systems" as a unifying umbrella "for sensing, simulation and digital models/shadows/twins" [12], which has the merit of keeping the distinctions visible while acknowledging that real deployments mix levels across subsystems.

Standardisation has taken a different route. The ISO 23247 series defines a digital twin in manufacturing as "a fit for purpose digital representation of an observable manufacturing element with synchronization between the element and its digital representation" [13]. "Fit for purpose" moves the criterion from data-flow topology to intended use, and "synchronization" is left deliberately unquantified. Architecturally this is permissive where Kritzinger is strict, and Section 8 returns to the tension.

### 3.2 Dimensional decompositions and their architectural residue

Manufacturing research supplied the field's other founding abstraction: the enumeration of a twin's constituent dimensions. The three-dimension model — physical entity, virtual entity, and the connection between them — was extended because it "cannot accommodate the new requirements from applications, technology, modeling objects", giving a five-dimension model that adds digital twin data and services [14]. The extension has not stopped; six-dimensional modelling has since been proposed for complex industrial systems [15].

These are conceptual decompositions rather than architectures — they say what a twin is made of, not how the pieces are deployed, who owns which interface, or what happens when one piece fails. But their residue is visible throughout the architecture literature, because most domain reference architectures are recognisably an elaboration of them. Data and services are promoted to first-class dimensions precisely because, in practice, they are where the software engineering effort concentrates. A survey of enabling technologies is explicit that services are the delivery vehicle and that "virtual models, data, and services are not isolated. They constantly interact with each other through connections among them towards collective evolution", requiring "application software, platform architecture technology, service oriented architecture (SoA) technologies, and knowledge technologies" [16].

The continued proliferation of dimension counts is worth reading as a finding rather than a curiosity. A field that has settled its reference model does not keep adding dimensions to it.

### 3.3 Reference model, reference architecture, architecture

Software architecture distinguishes a reference model — a division of functionality together with data flow — from a reference architecture, which maps that functionality onto a structure of components, and from a concrete architecture designed for a particular system. The mapping study applied this taxonomy to the digital twin field and found that 56.42% of primary studies propose a reference model [1]. The authors interpret this as a sign that architectural solutions "are still evolving and are just presented as simplified models" [1].

The consequence is a persistent gap between what is published and what is buildable. A recent effort to synthesise "a detailed reference model for DTs that reduces the concept-implementation gap" opens by stating that "existing definitions and reference models of digital twins are overly abstract, impeding their" application [17]. Even careful reference architecture work reports the same tension from the inside: a reference architecture for twin-based predictive maintenance, built with domain analysis, feature modelling and UML views, notes that "using high-level modules caused the 5C architecture to be rather generic" [18]. Tooling has begun to attack the gap from the other end. A template-based modelling method builds reference architectures out of twin templates, each abstracting a family of similar services, wraps existing services as modules that realise those templates, and then derives concrete architectures with a product-line configurator [19]. The reference-to-concrete step becomes a configuration choice rather than a redesign.

There is a counterweight. Expert interviews conducted as part of the mapping study found that reference models are valued by industry for economic reasons: "it is easier to assess certain qualities using reference models or architectures, which require a lower amount of time and money", and reference models "are actually desirable solutions from a company's perspective since they are economically cheaper to leverage for multiple business and technical analyses" [1]. The abstraction that frustrates implementers is what makes the artefact reusable for evaluation. Both readings are correct; they simply serve different stakeholders, which is itself an architectural observation.

## 4. The architectural design space

### 4.1 Patterns and their measured frequencies

The digital twin field's pattern usage has been measured, which is unusual and valuable. Across 140 primary studies, "the most recurring pattern is the layered pattern (35.9%), followed by SOA (29.3%), component-based (13.8%), and microservices (6.6%)", with publish-subscribe at 5.5%, event-driven at 2.7% and pipeline at 2.2% [1]. Ten unique patterns were identified in total, and patterns are usually combined rather than used alone: 38.19% of studies use more than one, and "the layered and SOA combination is, by far, the most used combination and accounts for 27 studies" [1].

Two things stand out. The first is the dominance of layering, which the study attributes to its role "as an interface between physical and virtual space, consolidating syntax and semantics of different perspectives" [1]. Layering is the natural response to a system whose components differ in their rate of change and their technology stack. The second is the low share of microservices, which contrasts sharply with mainstream distributed systems practice. The study's expert interviews offer an explanation that is architectural rather than fashion-driven: microservices "may be less used because of the rigidity of some domains

that can be safety-critical" and demanding of certifiable solutions "which are harder to build using given patterns" [1]. A twin embedded in a certified control loop cannot freely adopt a style whose defining property is independent, frequent, uncoordinated deployment.

The service-oriented share deserves a note. Service orientation reached digital twins partly through Industry 4.0, where the twin is the mechanism that makes service-based production possible: work at Fraunhofer derives "the minimal data to be contained in digital twins to enable an Industry 4.0 service-oriented architecture", describes services by "their capabilities and their quality of service in terms of time, money and resulting product quality", and validates the models by implementing the use cases "on two model plants" [20]. Here the twin is not a component within an SOA; it is the artefact that makes the SOA describable.

### 4.2 The layered stacks, and why they disagree

If layering dominates, one would expect convergence on a set of layers. There is partial convergence and instructive divergence.

The mapping study found that most layered digital twin architectures use four layers: "physical, data and network, virtual, and application layer", where "the physical layer is where machines, sensors, and actuators reside" and connects upward to data and network [1]. This is the modal stack, and it is essentially the five-dimension model with connection and data merged into a transport-and-storage tier.

Other stacks compete. A reference architecture for the Industrial Internet is "composed of three interdependent layers (i.e., physical layer, DT layer and DT networks layer) and four critical attributes (i.e., privacy, security, awareness and real-time)" [21]. This is a materially different decomposition: the twin is a single layer, and a further layer is devoted to networks *of* twins. Treating privacy, security, awareness and real-time as cross-cutting attributes rather than layers is also a distinct choice, and arguably a better-founded one, since none of them can be localised to a tier.

The predictive maintenance reference architecture instead adopts the 5C layering — Connection, Conversion, Cyber, Cognition, Configuration — where the connection layer "provides the Physical Twin proxy, acquisition, storage, and communication", the conversion layer "provides security, data quality, and the predictive maintenance module", the cyber layer "contains sub-modules to create a simulation model at a certain abstraction level of the physical asset", cognition "provides visualizations and performs risk analysis", and configuration carries machine optimisation [18]. Yet another framework for predictive maintenance twins proposes only three layers: "data collection layer, state judgement layer, service decision-making layer" [22].

Three, four and five layers, with different names and different assignments of security and storage, all for overlapping problems. The disagreement is not sloppiness. It reflects genuinely different separation criteria: by physical/digital boundary, by data transformation stage, by ownership, or by rate of change. A designer choosing a stack should therefore choose the criterion first and derive the layers, rather than adopt a published stack whose criterion is implicit.

### 4.3 Pattern catalogues: the finer grain

Below the level of whole-system stacks sit catalogues of reusable solution structures. The most developed is a systems architecture design pattern catalogue containing nine patterns — Digital Model, Digital Generator, Digital Shadow, Digital Matching, Digital Proxy, Digital Restoration, Digital Monitor, Digital Control, Digital Autonomy — each described with "a well-defined pattern documentation template" and positioned against phases of the systems engineering lifecycle, illustrated through a multi-case study in agriculture and food [23]. The catalogue is valuable because it converts the integration-level taxonomy of Section 3.1 into design vocabulary: Digital Shadow, Digital Monitor, Digital Control and Digital Autonomy are a ladder of increasing authority, and naming them separately makes the authority question explicit at design time. The predictive maintenance domain analysis found that "most DT-based PdM systems apply the Digital Monitor design pattern" [18] — that is, the field's most industrially mature application sits deliberately below the control rung.

A second catalogue addresses a specific and underappreciated deficiency of commercial platforms. Platform modelling capabilities "are mostly concerned with structural viewpoints, i.e., providing an overview of available components including their current and historical sensor values", while "behavioral viewpoints did not yet receive much attention on these platforms" [24]. The catalogue supplies modelling patterns for augmenting twin models with behaviour "without requiring any extension of the code bases of Digital Twin platforms", demonstrated on Microsoft's platform in an additive manufacturing use case [24]. That a workaround catalogue is needed at all is a finding about platform architecture: the dominant platforms model state well and behaviour poorly.

A related concern is structural correspondence. Work on declarative lifecycle management, classified under software architectures, argues that structures "of the physical system should be matched correctly in the digital twin by the corresponding DT components" [25]. Correspondence is easy to state and hard to preserve under change, which returns us to evolution.

### 4.4 Deployment: cloud, edge, and the continuum

Deployment is where digital twin architecture becomes concrete, and where the sharpest disagreements are found.

The cloud-centric position has the longest history. An early reference model, C2PS, was explicitly "a digital twin architecture reference model for the cloud-based cyber-physical systems", identifying "various degrees of basic and hybrid computation-interaction modes" and using "a Bayesian belief network" for its smart interaction controller so that "the system dynamically considers current contexts" [26]. Cloud deployment gives elasticity, managed storage, and a single place to aggregate across assets.

The counter-position is stated bluntly by work on edge digital twins. The Edge Digital Twin architectural model enables "the lightweight replication of physical devices providing an efficient digital abstraction layer", with state captured "through a dedicated physical adapter" and outgoing actions translated and forwarded so that "the two counterparts [are] always bidirectionally synchronized" [27]. Crucially, the same work argues that porting a cloud-native twin to the edge does not solve the problem: "even if it is theoretically possible to deploy some cloud-oriented solutions (e.g., Eclipse Ditto) also on the edge, there

is the concrete risk to introduce unnecessary delays and force the overall architecture to a monolithic vision of DTs" [27].

The mainstream resolution is hybrid, and it is argued from a specific trade-off: a cloud "architecture has sufficient computational resources, but it also exposes inherent poor latency performance", so "hybrid cloud-edge computing architectures" are needed together with "a near-instant and high-throughput communication network for synchronization" [28]. The continuum framing makes placement a first-class variable, and work on the digital twin continuum ties placement directly to two coupled qualities: twin schemas are "intrinsically linked to how accurately the DT represents the corresponding PA, i.e., its fidelity" and to "the communication latency between the DT and the Application", with the possibility of "multiple packages to cater to the same schema" on different platforms [29]. That last clause is the architecturally interesting one: it makes fidelity a deployment-time selection rather than a design-time constant.

Serverless has entered as a further option. A microservices-based and serverless-ready twin model is proposed as a foundation "for cost-effective DT deployment and orchestration", on the argument that the combination addresses "accommodating variable application requirements, managing load imbalances, and mitigating network faults" [30]. Serverless twin platforms built on Kubernetes have been targeted specifically at mid-sized cities, on cost-accessibility grounds [31]. Placement in the network has also been treated as an architecture question in its own right, with dedicated analysis of "possible architectures for placing a DT in the network" given that twins "require significant resources for communications, computations, and data storage" [32].

### 4.5 Composition: from single twins to ecosystems

Almost all the foundational literature describes a twin of one asset. Almost all interesting applications need many. Composition is therefore the design space's fastest-moving region.

The most systematic treatment couples aggregation with service orientation. A design framework proceeds through "1) needs and constraints analysis, 2) physical system decomposition, 3) services allocation, 4) performance and quality considerations, 5) implementation considerations and 6) verification and validation", working against "a general reference architecture that combines a digital twin aggregation hierarchy with a service-oriented architecture", which "allows for the separation of concerns, computational load distribution, incremental development and modular software design" [33]. The pairing is deliberate: the hierarchy gives the decomposition, the service layer gives the coupling mechanism.

A concrete manufacturing instance shows the hierarchy in operation, with "monitoring modules in the local digital twins of the workstations, whereas the production rate is evaluated and assured by the global digital twin of the shop floor", supporting "decentralized and integrated decision-making for re-scheduling" [34]. Manufacturing surveys supply matching vocabulary — unit level, production line, shop floor, system-of-systems level [35] — and the scaling argument is explicit: "The spatial span of the complex digital twins may be huge, and it is impossible to express all the information on the same scale, so the scalability of complex digital twins is the ability" to span scales [36].

The most ambitious reframing is the Web of Digital Twins. It begins by naming the limitation: "The dominant view developed in the literature so far is about the virtualisation of individual physical assets in a closed-system perspective", against which it proposes "an

open, distributed and dynamic ecosystem of connected digital twins, functioning as an interoperable service-oriented layer for applications running on top", validated through case studies in healthcare and smart mobility [37]. Related work supplies the naming machinery: in distributed twin systems "services are identified by a pair (service type, physical entity), where the same type of twin can be associated with multiple physical entities" [38].

Composition also has a governance side that a single-asset twin never meets. Work on twin systems-of-systems observes that twins are currently assembled ad hoc and bottom-up, and separates two problems: horizontal integration of one twin's own heterogeneous parts, and vertical composition of whole twins into a larger system, where conflicts between their data, models and services have to be resolved [39]. An architecture for composite twins that span organisations lists trust, governance, ownership, security and privacy among the requirements it has to meet [40]. An industry consortium's platform framework makes the twin system, rather than the twin, the unit of composition: a system of systems whose purpose is to realise a twin, with integration interfaces, synchronisation, and trust and governance across system boundaries as the concerns of composable and federated twins [41]. The Web of Digital Twins has since been built as a hypermedia layer in which every twin offers the same uniform interface over its local platform, and consumers query the whole ecosystem as one navigable knowledge graph [42].

Nonetheless, a gap remains acknowledged: reviewing composition for Internet-of-Everything applications, one study reports "a notable absence of architectures that support" composition of heterogeneous twins [43]. Composition is well motivated and unevenly supported.

## 5. Quality attributes and architectural tactics

Architecture exists to achieve qualities, and the interaction between multiple qualities is the substance of architectural analysis [44]. The digital twin literature has enough measured evidence to discuss these interactions concretely rather than aspirationally.

### 5.1 Maintainability and evolvability

Maintainability is, empirically, the field's leading concern: it is "the most mentioned quality attribute (16.2%)" across the mapped primary studies [1]. The study also had to extend ISO 25010 with three attributes mentioned in more than ten studies each — scalability, reconfigurability and extensibility [1] — all of which are, in effect, maintainability under different time horizons.

The dominant tactic set is imported from mainstream software engineering. DevOps applied to twins has produced explicit frameworks structured around "(1) Software methodology to support the systematic and iterative development and evolution of DTs; (2) Microservice architecture to enable the independent development and composition of different aspects of DTs; and (3) DevOps infrastructure to support the development, integration, and deployment of DTs" [45]. TwinOps adds model-based engineering to the same loop, combining "systems engineering models to capture system requirements, interfaces, and the system's functional decomposition into subsystems", simulation models "to evaluate the system general behavior", and "engineering models for the system's design, and then model transformations towards analytical models" [46], [47]. Treating model transformation as

a pipeline stage is the key move: it makes model consistency a build-time property rather than a matter of discipline.

Evolution has to reach the running twin's state as well as its code. A fluent-API approach treats a twin's schema, its current snapshot and its historical data as three levels that must co-evolve whenever a change in the physical system triggers a change in the twin [48]. GitOps has been mapped onto the twin's own runtime lifecycle operations, such as connecting, synchronising and configuring, so that a deployed twin is held in version control and can be redeployed without interruption [49].

Model-driven engineering for twins has become a community position, with a multi-institution communication setting out its opportunities and challenges [50]. Runtime observability of the architecture itself has also been proposed, on the argument that "stakeholders require tools to understand the behavioral state" and need "systematic and automated understanding of the data flow between DT components" [51]. Meanwhile the reproducibility of twin research is itself under scrutiny [52], and reporting frameworks have been proposed to make twin implementations comparable, noting that "composition of interfaces, are current challenges in relation to DT model-driven engineering methods" [53].

### 5.2 Latency, synchronisation and fidelity

Fidelity and synchronisation rate are not implementation details; they are definitional. A twin "is characterized by a symbiotic relationship between a physical entity and its virtual representation" and by "its fidelity, rate of synchronization, and choice of" modelling [54], and a network-oriented survey defines a twin as a representation "synchronized and updated from real-time data, at a specific frequency and fidelity" [55]. Fidelity is also not one property: a review of twins in engineering design separates model fidelity, data fidelity and communication fidelity, each bearing on accuracy and synchronisation [56]. Making these architecturally tractable requires measurement, and a recent survey of twin-as-a-service platforms names "twin fidelity, twinning rate, and inference latency" as the metrics "critical for evaluating the performance of individual components". It finds the literature short of metric-oriented approaches that link platform design to measurable behaviour, and faults an earlier survey of twin development tools for discussing fidelity, granularity and synchronisation latency without providing "concrete numerical benchmarks for these metrics" [57]. Runtime measurement has begun to appear: one prototype aligns the behavioural traces of a twin and its physical counterpart over sliding windows to measure behavioural fidelity and detect delays and deviations, demonstrated on elevator traces [58].

The most architecturally significant treatment concerns what happens when synchronisation fails. Rather than assuming a reliable channel, work on time discrepancy handles the case where "a DT and its PT get out of sync as a result of disturbances in the normal operational conditions of the DT-PT system, e.g., due to network degradation or temporary network drop", with the explicit goal of providing "a best-effort functionality" [59]. The coupling decision underneath this is made explicit in work on autonomous reconfiguration. A tightly coupled twin-enabled system, in which the twin makes the decisions, stops operating when the twin is unreachable; a loosely coupled one keeps operating on local decision mechanisms and resynchronises with the twin when the connection returns [60]. Designing for degraded synchronisation rather than for its absence is a mature architectural stance. It remains uncommon in the corpus, although the mobility case of Section 6.3 applies it.

### 5.3 Interoperability

Interoperability is where digital twin architecture meets standardisation, and the news is mixed. The Asset Administration Shell is the most institutionally supported information model, specified as the vehicle for "the exchange of information between partners in the value chain" [61], with open-source implementations of its reactive form surveyed in detail [62] and its three types analysed specifically for their "feasibility for digital" twins [63].

But multiple description formats coexist and do not agree. The problem is stated plainly: descriptions such as AAS documents "can enable collaboration between machines by presenting their properties in a standardised format. However, the existing DTD formats present information in an incompatible way, hindering interoperability" [64]. The field's response has been bridging rather than convergence. One study builds "a mapping between the Digital Twin Definition Language (DTDL) and Asset Administration Shell (AAS)" and implements and tests the transformation "for a physical asset" [65]; another maps DTDL to OPC UA [66]. Comparative metamodel analysis across Azure's DTDL and the Eclipse Hono/Vorto/Ditto suite has required "an UML extension as pivot for bridging the gap" [67]. An industry consortium framework has attempted to raise the discussion above formats, organising around seven interoperability concepts beginning with system-centric design and built on the notions of digital thread and system-of-systems [68].

Behavioural interoperability is a separate problem, addressed by co-simulation standards. Co-simulation "prescribes the communication interface and protocol between the simulation units and orchestrator", with FMI, HLA and DEVS as the established standards [69]. The necessity is structural: models span "diverse formalisms, some describing discrete digital processes and some describing continuous flows", and "the coupling of such models requires a standard in order to ensure interoperability" [70]. Tooling has not kept pace — "while standards like Functional Mock-up Interface (FMI) facilitate interoperability, there remains a lack of flexible, research-oriented tools that support rapid prototyping" [71] — and open-source twin platforms have a well-documented blind spot here: projects such as Eclipse Ditto and iTwin.js "focus primarily on facilitating the creation and management of twins and their associated data, but do not provide adequate support for simulation" while other projects "concentrate solely on simulation" [72]. Broader comparative surveys of fourteen open-source twin frameworks across ten dimensions confirm that combining tools is usually necessary [73], and platform proliferation across sectors continues at uneven maturity [74].

### 5.4 Security and trust

Security in digital twins is not a component; it is a property of the whole structure, and the literature says so. A comprehensive threat survey classifies threats "taking into consideration its functionality layers and the operational requirements", insisting that "any security analysis of a DT must take into account the four functionality layers", and labels each threat by the layer it originates in [75]. Its most architecturally useful contribution is the cascade analysis: "a threat [T1.1] in Layer 1 may involve a synchronization variation that implies significant changes in the final management of the digital models included in Layers 2 and 3" [75]. A sensing-layer integrity failure propagates upward as a model-validity failure, which is precisely the failure mode a layered architecture is supposed to contain and does not.

Deployment style changes the risk profile: “the more decentralized, the lower the risk of bottleneck or exhaustion” [75]. Other reviews reinforce that “all the components encompassing the infrastructure must be” protected and that risk trade-offs must be assessed, “such as latency and feasibility issues, especially when dealing with specific systems like those in the energy sector” [76]. Tactically, detection must complement prevention: “attack detection is a critical component of a robust DT cybersecurity strategy in addition to prevention and mitigation measures like firewalls, encryption and access controls”, covering “the CPS including the DT, communication channels, sensors, actuators” [77]. Standards context is available — ISO 23247 alongside “ISO 8000 for data quality and ISO 27001 for organizational security” [78] — and where twins span organisational boundaries, zero trust architecture is the relevant reference model from the security side [79]. Twins are also used as security instruments rather than only as security liabilities, with security simulation frameworks tested on “a PoC of an industrial filling plant and a conveyor belt” while evaluating “system performance and latency” [78].

### 5.5 The trade-offs, stated

The single most useful trade-off result in the corpus is that “the high reliability, maintainability and on-demand scalability offered by microservices can lead to increased latency, and in this case, a trade-off between performance and other QAs is needed” [1]. This is the central tension of digital twin architecture in one sentence. The tactics that make a twin maintainable over its asset’s lifetime — decomposition into independently deployable services, indirection through message brokers and gateways, abstraction of storage behind adapters — all add hops, and hops cost latency, and latency is a defining quality of a twin. The corpus also links pattern choice to quality: microservice patterns are mentioned together with maintainability (19.2%), compatibility (19.2%) and scalability (23%) [1]. The five case studies in the next section can be read as five different resolutions of exactly this conflict.

## 6. Five example architectures

### 6.1 Manufacturing: makeTwin, a platform reference architecture

**Drivers.** The stated motivation is a gap in the platform market rather than in a single factory. Commercial twin platforms are judged inadequate from two directions: researchers find them insufficient for theoretical investigation, while for enterprise users the platforms are “highly specialized, providing professional-grade multi-physical coupling analysis, diverse algorithms, and advanced 3D visualization development techniques” yet leaving “a significant challenge for common enterprise users who struggle to customize digital twin application service systems to align with their specific requirements”, from which the authors conclude that “a configurable, modular, and universally applicable digital twin development platform” is required [80]. The architecture is therefore a product-line architecture: its purpose is to make many twins cheaply, not to make one twin well.

**Structure.** makeTwin “is comprised of ten fundamental functional modules, namely twinModelBuilder, twinDataProcessor, twinAlgBuilder, twinIoTConnector, twinInteractor, twinSimulator, twinLibrary, twinVisualization, twinSceneTemplate, and twinAppDeployer”, where “each module provides interoperability with features such as configurability, callability, modifiability, and extendibility” and “the eight core functions of makeTwin (excluding twinAppDeployer and twinLibrary) can interact through customizable API” [80]. The ex-

clusion of two modules from the API mesh is telling: twinLibrary is a shared repository and twinAppDeployer is a build-and-ship function, so both sit outside the runtime interaction graph by design.

Internal decomposition is deep. twinModelBuilder splits into ModelContainer (import, creation, conversion, lightweighting) and ModelIntegration (assembly, fusion, validation, correction), where model assembly covers "individual models, unit-level models, system-level models, and complex-system-level models in the spatial dimension" including "the creation of topological and constraint relationships", and model validation entails "verifying the consistency between digital model output and physical object output" [80]. This is the aggregation hierarchy of Section 4.5 realised inside a single module. twinIoTConnector carries the physical coupling, addressing "the digital twin guidelines of accuracy, real-time, consistency, security, and reliability" through protocol selection, creation and configuration, physical data source access, system data aggregation and data asset management [80].

**Variability and deployment.** Reuse is explicit rather than incidental. twinSceneTemplate provides "a collection of common scenario templates across various fields" — smart equipment, shop-floor logistics, stereo warehouse, smart energy — which can be imported at project start and saved back after development [80], and the documented workflow branches on whether an applicable template exists before falling back to configuration from the component libraries [80]. twinAppDeployer supports both offline deployment, where "the digital twin application system is installed locally after configuring the environment on a designated computer or server", and online deployment to "a cloud server environment" accessed via web links [80].

**Validation and limitations.** The architecture is instantiated for a chemical fibre textile shop floor, with three functional requirements — three-dimensional visualisation of production logistics, high-fidelity modelling of the winding, packaging and warehousing units, and adaptive scheduling optimisation [80]. The authors state that "the architecture and functions need to be further iterated and improved to forming an applicable engineering software, as well as ensuring the adaptability to various industrial and application scenarios" [80]. In architectural terms: the variability mechanism has been demonstrated on one product, not across a family, so its central claim remains partly unvalidated.

**Standard comparison.** makeTwin can be read against ISO 23247, which decomposes the same domain into entities rather than modules: a user entity hosting PLM, ERP, MOM and human-machine interfaces; a digital twin entity with operation-and-management, application-and-service, and resource-access-and-interchange sub-entities; a device communication entity with data-collection and device-control sub-entities; and a cross-system entity providing "common functionalities such as data translation, data assurance, and security support" [13]. The cross-system entity is the structural counterpart of the cross-cutting attributes of Section 4.2, and its existence concedes that some concerns will not localise. The standard has been instantiated in use-case scenarios implementable "at the National Institute of Standards and Technology (NIST) Smart Manufacturing Systems (SMS) Test Bed or on any manufacturing shop floor that has similar maturity level of digital transformation" [81], and it presumes integration with incumbent systems rather than their replacement [82]. Conformance is not automatic: experience in industrial projects suggested that the standard is not completely followed, which led to an analysis of how far the twin architectures documented in the literature align with its reference architecture [83].

### 6.2 Healthcare: an ecosystem of twins for operating room management

**Drivers.** This system is an integration problem before it is a twin problem. The legacy QR-code scanning system "stores the data about each hospital and its ORs in a local relational database that aggregates data coming from modules of the healthcare information system", and "it was unfeasible for this project to change the data storage on the original legacy system" [84]. The dominant force is therefore an immovable constraint, and the architecture is organised around routing past it. The clinical need was equally specific: medical staff "identified data inconsistency as the main issue to be solved as a first step" [84].

**Structure.** Three tiers. First, a data integration and standardisation tier: "a separate centralised component that is responsible for polling the databases and integrating the data coming from the different hospitals", writing into "an HL7 FHIR server based on the IBM FHIR server implementation" [84]. Surgery is mapped onto three linked FHIR resources — an appointment defining "the nature of the planned surgery, its expected duration and its priority", an encounter for patient–staff interaction, and multiple procedures "representing each action performed on the patient during the overall surgery process" — such that "the duration of each step of the perioperative process are kept in the appropriate resource" [84].

This tier is an anti-corruption layer in the classical sense, and the authors justify it in exactly those terms: the standardisation "decouples the new DT layer from the legacy systems involved, allowing to build DTs with a modern approach and keeping them independent from the evolution of the healthcare information system" [84]. They also predict its own obsolescence — "we can expect this step to not be necessary for future applications once the standards reach a pervasive use" [84] — which is an unusually explicit statement that a component exists to absorb a temporary environmental deficiency.

Second, the digital twin layer, containing a Surgery DT and an OR-DT. The Surgery DT "periodically queries the data from the implemented FHIR server … and aggregates the data from the different resources into a Surgery entity model that implements the rules defined above and generates the appropriate warnings, decorating the data coming from the FHIR server with the new information" [84]. Encoded rules cover missing or misordered timestamps, anomalous surgery or step durations "depending on the average time defined in medical literature", anomalous ward-to-OR transfer duration, and the integrity constraint "that only one surgical procedure can be performed at the same time in a single OR" [84]. The OR-DT holds static location information and computes current availability by checking for an open encounter with no registered end time [84].

Third, a management dashboard web application on top of the twin layer, with authenticated roles separating viewing from editing because "only the authorised leader of the medical staff should be able to annotate" a surgery [84].

**A general design principle, stated locally.** The twin layer exposes data in two forms: "either directly as FHIR resources or, in a more compact format that aggregates the relevant details about each step in a single Surgery resource, including the warnings computed by the DT". The authors generalise: "This duality of data representation is a common feature of DTs… DTs should not be built with a single application in mind, but they should be seen as a way to store data concerning a physical asset in a coherent way" [84]. This is a clean statement of why a twin is an architectural layer rather than an application backend.

**Limitations.** Stated frankly. "The DT Layer implements a partial view of the vision", and

because the system "still requires a human input to label each step with the current time of execution, the real-time acquisition is not guaranteed", with IoT instrumentation deferred to future work [84]. The polling-based coupling is also acknowledged as provisional: "we can imagine that in the future the DT could listen for events concerning the update of resources in the FHIR server" [84]. On Kritzinger's scale this is a digital shadow with a partly manual acquisition path [11].

**Intra-domain contrast.** Healthcare twins do not converge on this shape. A hospital ward-flow twin is built on entirely different foundations, "leveraging executable formal models for system exploration, ontologies for knowledge representation and an SMT solver for constraint satisfiability", turning "a stream of arriving patients ... into a stream of optimization problems" [85]. Others adopt a cloud-centric framing for elderly care services [86]. The scoping literature confirms the breadth of ambition without confirming architectural convergence [87].

### 6.3 Mobility: a cloud–edge mobility digital twin

**Drivers.** Three classes of entity must be twinned at once — vehicles, humans and traffic — and the data is generated by a mobile, intermittently connected fleet. The architecture is explicitly an instantiation rather than a proposal, built "to realize the Digital Twin concept in the real world" [88].

**Structure.** Four layers: "1) the cloud layer, which is built on AWS and its Virtual Private Cloud (VPC); 2) the edge layer, which has a storage component; 3) the device layer, which generates data; [and an] API layer, which hooks up the cloud layer with external" sources [88]. The mapping between conceptual framework and physical deployment is stated rather than left implicit: the physical space "is represented by the device layer", the communication plane sits "within the edge layer's communication component", and the digital space "spans the whole part of the cloud and API layers, as well as part of the edge layer (except for its communication" component) [88]. Making this mapping explicit is good architectural practice and comparatively rare in the corpus.

Within the cloud layer, storage is polyglot by design: "Amazon S3, a scalable storage infrastructure to build our Digital Twin data lakes", "Amazon DocumentDB" with MongoDB compatibility for "flexible, low latency queries to obtain a near real-time record of events in parallel on a massive scale", and Redis [88]. AWS IoT Core provides device connection, a rule engine "evaluates the rules configured for entities (e.g., humans and vehicles) on the data received", data workflows are "triggered via Apache Airflow", and identity uses "a simple identity layer on top of the OAuth 2.0 authorization protocol" [88]. The API layer admits external traffic and map data and supports further microservices [88].

**The measurement that matters.** This case study reports end-to-end latency, which makes it unusually assessable. Uplink time scales with batch size, "with a minimum of 1.4 s for 12 MB and a maximum of 21.1 s for 240 MB", but "the majority of the end-to-end latency is made up by the aforementioned cloud computing processes, which roughly takes 16 s" [88]. The architecture does not put those sixteen seconds inside a control loop. In its personalised adaptive cruise control case study, the cloud builds a driver-specific model that the vehicle then uses; the 16 s is the warm-up before that model is ready each time the driver switches the feature on, and a baseline PID controller keeps car-following safe in the meantime [88]. Real-time control stays on the vehicle, the cloud supplies what the controller runs, and a

local fallback covers the gap: the loosely coupled arrangement of Section 5.2 [60], used as a concrete answer to the quantified latency cost of the maintainable, elastic style [1]. The reported benefit is behavioural: drivers took over from the personalised controller 86.4% less often than from a conventional PID-based one [88].

**Limitations.** Named clearly. Standardisation "can face numerous challenges, since the consensus may be difficult to reach across the public sector (e.g., transportation agencies) and the private sector (e.g., automotive manufacturers, suppliers, and network providers)" [88]. Vendor coupling is acknowledged and bounded — the framework is deployed on AWS, but "this does not necessarily mean our proposed MDT framework can only work on AWS", with Azure, GCP and Alibaba Cloud named as alternatives [88]. And the trust problem is stated squarely: "The nature of public cloud platforms inevitably gives away the control of resources to some extent, which may introduce cybersecurity and privacy risks for end users", particularly for Human Digital Twin microservices [88]. Vehicle mobility also stresses the connectivity assumptions, with DSRC widely deployed but limited and C-V2X "not affordable and widely deployed" by comparison [88].

**Domain contrast.** An alternative urban architecture rejects this system's automation premise outright. A Digital Twin Federation for urban mobility observes that twins "have been proposed to automate solutions, yet such automation often conflicts with the requirements of human-driven planning in urban mobility systems", and is therefore "designed for comprehensive urban mobility assessments with human-in-the-loop control", integrating "multilevel simulations with real-time data ingestion, traffic management and optimisation, advanced data analysis, and interactive visualisation" [89]. A smart-city twin architecture built for the edge-cloud continuum targets "dynamic resource management, real-time data processing, and autonomous decision-making" and is demonstrated on autonomous traffic management in Issy-les-Moulineaux [90]. Federated city twin efforts have also produced shared environments allowing "models, data, and simulations to" interoperate, against a background where prototype city twins "may not be interoperable without consistent guidelines to follow" [32]. Governance is a real constraint here, since these technologies "have the potential to reshape or redefine how we think and manage city governance and our key or critical urban infrastructure" [91].

### 6.4 Energy: a digital twin for a floating offshore wind turbine

**Drivers.** This case is the clearest instance of physics driving structure. The purpose is "to follow the life cycle of a wind turbine in real time and ultimately provide tangible signals to inform O&M decisions", under a hard architectural constraint chosen deliberately: "Our goal is to achieve this by relying only on measurements expected to be available on most wind turbines, thereby avoiding the extra cost of adding sensors" [92]. Sensing is treated as an economic budget rather than an engineering free variable, and everything else follows. Whether additional optimally placed sensors would repay their cost is left explicitly open [92].

**Structure.** A pipeline rather than a layered stack. Inputs are "high-frequency measurements from the SCADA system (e.g., power, pitch, rotor speed)" [92]. The core is a state estimator: "The core algorithm in the estimation is a Kalman filter that uses a linear wind turbine model", estimating "aerodynamic states (wind speed, thrust) and motions of the structure (e.g., surge, pitch, tower deflection)" [92]. Estimated states then feed "a virtual sensing step to produce quantities of interests (QoIs), such as the loads at key locations of

the structure", which "are then intended to be postprocessed by a monitoring and diagnostic tool to generate the data needed to perform condition-based O&M" [92]. Deployment intent is stated: "The digital twin is intended to run in real time on a cloud platform" [92].

The model supply chain is itself an architectural element. Linear models are obtained through two independent routes — "a suite of dedicated Python tools implemented as part of this work and the OpenFAST linearization feature" — and "in our final version of the digital twin, we use components from both approaches" [92]. Maintaining two derivations that must agree is a validation tactic: cross-checking substitutes for an oracle that does not exist. Validation proceeds in two stages, first against "numerical experiments" where "the QoIs are directly accessible and can be compared to the estimates for verification purposes", then against the TetraSpar floating prototype, a 3.6 MW turbine with a 130 m rotor equipped with additional sensors used only to validate the estimates [92].

**Limitations.** Exceptionally well demarcated. "Structural health modeling, monitoring and diagnostics, and O&M decisions" are "postponed to future work, even though they are essential steps to achieve our final vision", and also outside scope are "the use of historical data to assist in the diagnostics, the use of estimates to perform model updating, and real-time implementation" [92]. The last item is significant: the architecture is designed for real-time cloud operation but has not been operated that way. Tool coverage is partial too — in the in-house library "currently, no controller or aerodynamic module is present" to "cover the full range of options available with OpenFAST" [92]. This is a digital shadow with a rigorous estimator, honestly labelled.

**Domain contrast.** A diagnostic digital twin for the same asset class defines a capability tier explicitly: "a virtual representation of an asset that combines real-time data and models to monitor damage, detect anomalies, and diagnose failures, thereby enabling condition-based and predictive maintenance", while noting that "the DNN architecture, learning rate, and batch size have been manually tuned" [93] — manual tuning being a maintainability liability in a system meant to run for an asset's lifetime. Structural test platforms underpin this work physically, such as a 12.6 m reference blade and its moulds developed as a research and demonstration platform [94]. Predictive maintenance twins cluster in this domain: within energy, "most research is focused on Renewable Energy sources (n = 4), such as Wind Turbines", with smaller counts for fossil energy, grids and nuclear [95]. Grid-connected twins inherit an existing sectoral reference architecture: NIST's smart grid architecture consists "of several domains and subdomains", each reflecting "the purview of stakeholders, including the consumers, market operators, energy managers, utility services, generators, transmission operators, and distributors" [96]. Decomposition by stakeholder purview, rather than by data-flow stage, is a fourth separation criterion to add to those in Section 4.2.

### 6.5 Built assets: a microservice twin under a DevOps regime

**Drivers.** Here the driver is organisational as much as technical. Twins for built assets are multidisciplinary artefacts, and the framework's stated purpose is to let "domain experts" contribute "without having to worry about the details of the software engineering process such as testing, deployment, distributed systems considerations, and monitoring of the software contributions" [97]. The architecture is designed to partition expertise, not only computation. Twins here are treated frankly as "complex software systems" [98].

**Structure.** Three layers, presented "using a UML composite structure diagram": "DT

Data Management, DT Data Distribution, and DT Services, all three fueled by data from the underlying AT and its surrounding Environment", including external data sources [97]. The distinguishing commitment is that "all DT internal communications are made through the Pub/Sub Internal Data Distribution layer", with ports explicitly labelled "P" for publisher and "S" for subscriber and the topic named beside each port [97]. Naming topics in the architecture diagram makes the coupling structure inspectable, which most twin architecture diagrams do not.

Data sources are threefold: IoT devices ranging "from resource-constrained microcontrollers (e.g., Arduino-based setups) to more capable Single Board Computers (SBCs) such as RPi running lightweight containerized environments"; metadata, notably "the BIM model of the GRIDD Lab [which] offers a 3D spatial representation of the laboratory, embedding details about the physical layout along with many other 'as built' details such as sensors placement"; and external data such as an OpenWeather API feed [97].

Storage is heterogeneous by design — MongoDB Atlas on AWS for asset detection JSON, Azure TSI for temperature and humidity time series, BIM data from Autodesk Construction Cloud — and the architecture's key tactic is that it "introduces dedicated data access adapters for each storage source, rather than depicting the storage technologies themselves", so that "if a storage solution needs to be replaced (e.g., migrating from Azure TSI to another time-series database or switching BIM providers), only the corresponding adapter must be updated - leaving the rest of the architecture unaffected" [97]. The distribution layer above it "provides a unified abstraction for information sharing that shields the upper layer from the complexities of data acquisition, processing, and storage" [97]. Services "operate as independent microservices, adhering to a modular architecture that ensures loose coupling and high cohesion", "containerized using Docker" so each "can run consistently across different environments, whether on the cloud or at the edge", forming "an easily deployable build artifact" [97]. Remote communication avoids "proprietary or component-specific protocols", using "RESTful APIs, using JSON messages", with AWS API Gateway dispatching to Lambda for one aspect and an Azure IoT gateway brokering the other [97].

**The reported failure, and why it is valuable.** The two aspects were developed independently and then integrated, and the integration exposed a defect: "while the Thermal Comfort aspect leveraged a dedicated data distribution component, the Asset Management aspect lacked an equivalent layer, making its integration more tightly coupled with upstream components. This asymmetry revealed the need for consistent abstraction mechanisms across all DT aspects, not only to streamline development but also to support independent evolution of each aspect" [97]. The authors also generalise the lesson that integration requirements must be anticipated "from the very beginning" because such decisions "have a direct impact on data flow, versioning, and deployment processes" [97]. Uniformity of abstraction across peer subsystems is an architectural obligation, and the omission is only discovered at composition time.

**Limitations.** The twin is human-in-the-loop by construction: "control actions remain unautomated", with "a dotted arrow" representing "the manual user feedback loop between the UI and the physical asset", while automatic regulation of thermal conditions is envisaged for the future [97].

**Domain contrast.** Construction twin-as-a-service work sets an explicit open-standards requirement, mandating support for IFC models and designing "such that the communication

middleware can be replaced, e.g., with Eclipse Ditto, thanks to the IfcCockpitAPI which abstracts away low-level implementation details" [99] — the same adapter tactic applied to middleware rather than storage. Elsewhere, BIM-integrated monitoring systems have been built at LOD 400 for urban infrastructure maintenance [100].

## 7. Cross-case comparison

Table 1: The five example architectures of Section 6 compared: domain, core structural idea, deployment, integration level and stated limitations.

| Architecture | Citekey | Domain | Core structural idea | Deployment | Integration level | Stated limitations |
|---|---|---|---|---|---|---|
| makeTwin | [80] | Manufacturing (textile shop floor) | Ten configurable functional modules interacting via customisable APIs; scene templates as the variability mechanism | Offline local install or online cloud with web access | Twin-capable via twinIoT-Connector and twin-Interactor; case study emphasises monitoring, scheduling optimisation and visualisation | Architecture and functions "need to be further iterated"; adaptability across industrial scenarios not yet ensured; validated on one instance |
| OR management twin ecosystem | [84] | Healthcare (surgical suite) | Three tiers: FHIR standardisation as anti-corruption layer, Surgery DT and OR-DT with encoded rules, dashboard | Centralised server components over multi-hospital legacy databases | Shadow; partly manual acquisition | "Partial view of the vision"; "real-time acquisition is not guaranteed"; polling rather than event-driven coupling |

| Architecture | Citekey | Domain | Core structural idea | Deployment | Integration level | Stated limitations |
|---|---|---|---|---|---|---|
| Mobility Digital Twin | [88] | Mobility (connected vehicles) | Four layers — cloud, edge, device, API — with polyglot storage and rule engine; vehicle, human and traffic twins | AWS VPC with edge storage and device tier | Twin: a cloud-built personalised cruise-control model runs on the vehicle, with a PID fallback during warm-up | ~16 s cloud stage dominates end-to-end latency, absorbed as a per-activation warm-up; standardisation consensus hard; public cloud cedes resource control, raising security and privacy risk |
| Floating offshore wind turbine twin | [92] | Energy (offshore wind) | Estimation pipeline: SCADA input → Kalman filter over linear models → virtual sensing → QoIs → diagnostics | Intended for real-time cloud operation | Shadow; O&M decision loop closed by humans | Structural health modelling, diagnostics, O&M decisions, model updating and real-time implementation all outside current scope |
| GRIDD Lab twin (JuNo-OPS) | [97] | Built assets (laboratory building) | Three layers with mandatory internal pub/sub, per-source storage adapters, containerised microservices | Docker containers on cloud and edge; AWS and Azure gateways | Shadow with human-in-the-loop actuation | Abstraction asymmetry between aspects discovered at integration; "control actions remain unautomated" |

Four observations follow from the comparison in Table 1.

**Structural agreement is low; problem agreement is high.** No two of the five share a layering. makeTwin is module-and-API oriented, the operating-room twin is tier-oriented

around a standardisation boundary, the mobility twin is deployment-tier oriented, the turbine twin is a dataflow pipeline, and the built-asset twin is broker-mediated. Yet all five confront the same four problems: heterogeneous sources needing adaptation, model or rule logic needing somewhere to live, a return path that is either deferred or explicitly human, and an evolution story. The uniformity of the problem set alongside the diversity of solutions is the strongest argument in this survey for reference models being pitched at the problem level, which is roughly what the mapped literature actually produces [1].

**Adapters are the field's most portable tactic.** Storage adapters in the built-asset twin [97], physical adapters in edge twins [27], twinIoTConnector's protocol configuration in makeTwin [80], the FHIR mapping tier in the operating room [84], and replaceable middleware behind an abstracting API in construction [99] are the same tactic under five names. Given that heterogeneity is the field's defining condition [3], this convergence is unsurprising and probably the most transferable result available.

**Four of five stop short of automated actuation.** Only the mobility twin closes an automated loop back to the physical system, and it does so by handing a personalised model to the vehicle's own controller rather than by issuing commands from the cloud [88]. The operating-room twin needs human input even for acquisition [84], the built-asset twin leaves "control actions ... unautomated" [97], and the turbine twin explicitly defers the O&M decision stage [92]. This is consistent with the finding that only 18% of publications describe bidirectional twins [11] and with the predictive-maintenance field's concentration on the Digital Monitor pattern [18]. Practising architects should treat full bidirectionality as an aspiration in most domains, not a baseline.

**Latency budgets differ by orders of magnitude and are rarely stated.** The mobility twin's measured ~16 s cloud stage [88] would be irrelevant to a scheduling twin and disqualifying for a control twin that waited on it, which is why that twin keeps control on the vehicle and treats the cloud stage as a warm-up. Of the five, only that one publishes numbers. That is the deficiency the DTaaS survey names when it finds the literature short of metric-oriented approaches linking platform design to measurable behaviour [57].

## 8. Cross-source disagreements

Four disagreements in the corpus are substantive rather than terminological, and none is resolved.

**Is a twin passive data or an active component?** The passive reading treats the twin as a synchronised representation that applications query. The active reading is stated as a critique of named platforms: in Azure "DTs are conceptualized as centralized cloud entities integrated within a robust yet isolated framework, lacking the capability for integration with other DTs", while with Eclipse Ditto twins "are depicted as passive entities, deferring the responsibility for implementing their behavioral model and lacking active monitoring of their behavior and performance during execution" [101]. The same group elsewhere describes "a recent shift towards envisioning DTs also as active software components ... autonomous entities" [30], and the Web of Digital Twins vision presupposes twins active enough to serve as "an interoperable service-oriented layer" for multiagent systems above them [37]. The disagreement has teeth: if twins are passive, behaviour belongs to applications and the platform can be a database with a schema; if active, the platform must host and orchestrate behaviour, and the platform's failure to model behaviour becomes a defect

rather than a scope decision [24].

**Cloud-first or edge-first?** Cloud-centric reference models are long established [26], and the mobility case demonstrates what is achievable on a commercial cloud stack once time-critical control is left on the vehicle [88]. The edge position holds that migrating a cloud twin edgeward risks "unnecessary delays" and forces "a monolithic vision of DTs" [27]. The hybrid position treats this as a resolved trade-off between resources and latency [28], but "hybrid" only names the problem; the continuum literature is more useful in making fidelity and latency jointly determined by placement [29]. Security cuts across the same axis, since "the more decentralized, the lower the risk of bottleneck or exhaustion" [75] while the mobility case reports that public cloud "inevitably gives away the control of resources to some extent" [88].

**Must a twin be bidirectional?** Kritzinger's criterion is strict: without fully integrated bidirectional flow, the artefact is a shadow or a model [11]. ISO 23247 is permissive, requiring a "fit for purpose digital representation … with synchronization" and leaving synchronisation unquantified [13]. Both positions are defensible and they classify the same systems differently. Four of this survey's five case studies would be twins under ISO's definition and shadows under Kritzinger's. The practical consequence is that a project should state which criterion it is using, because the answer determines whether an actuation path, a command-authority model and the associated safety analysis are in scope.

**Are reference models an achievement or a symptom?** The mapping study's finding that 56.42% of solutions are reference models is read by its authors as evidence that architectural solutions "are still evolving" [1], and others argue existing reference models are "overly abstract, impeding their" use [17]. The same study's expert interviews found industry values reference models precisely for their cheapness in analysis [1]. This is less a contradiction than an unacknowledged split in audience, but the literature does not usually declare which audience a proposed reference model serves.

## 9. Gap analysis

**What the corpus covers thinly.** Several architecturally important topics returned little after query reformulation. *Architectural evaluation* is the most conspicuous: the corpus contains general software architecture treatments of definition, evolution and analysis [44] and abundant twin architecture proposals, but almost nothing applying structured evaluation methods to a digital twin architecture and reporting the result. The mapping study reports that solutions are validated with "experience, analysis and example" rather than by persuasion [1], which is encouraging about rigour but does not amount to a quality-attribute-driven evaluation practice. Re-searching found two partial exceptions. One proposes a multi-criteria framework that scores domain-specific twin platforms by weighting expert judgements over the functional components of the makeTwin reference architecture [102]; another analyses how far published manufacturing twin architectures align with the ISO 23247 reference architecture [83]. Both assess platform fit or standard conformance, not the trade-offs between quality attributes. *Cost and organisational economics* is nearly absent as an architectural driver, appearing only obliquely through SME accessibility arguments [103] and platform-cost motivations [31]. *Runtime observability of twin architectures* is represented by a single dedicated tool [51]. *Decommissioning and end-of-life* barely appears, despite the lifecycle framing being central to the field's self-description [5]. Where it does appear, as in

an exemplar that names decommissioning among the lifecycle stages a twin must adapt its own structure to, it motivates self-adaptation rather than an architecture for retiring the twin itself [104].

**What is described but not measured.** The field states quality attributes far more often than it quantifies them. Fidelity, twinning rate and inference latency have been named as the necessary metrics [57], but only one of the five case studies publishes latency figures [88] and none publishes a fidelity measurement against a stated target. A first method for measuring behavioural fidelity at runtime has been prototyped [58], so the shortfall is now one of practice more than of technique. Without such numbers, claims that an architecture is scalable or real-time are unfalsifiable, and cross-architecture comparison — the point of a reference architecture — remains impossible. This connects directly to reproducibility concerns raised in the corpus [52] and to proposals for systematic reporting [53], including a characterisation scheme for twin case studies that asks each one to state its time-scale and its fidelity considerations [105].

**Where the standards story is incomplete.** Information-model standards are well developed and mutually incompatible [64], and the field's response has been pairwise bridging [65], [66] rather than convergence, with each platform contributing another domain-specific language [67]. Behavioural interoperability through co-simulation is better standardised [69], [70] but poorly integrated into twin platforms, several of which support data management without simulation or simulation without data management [72]. No source in the corpus proposes an architecture in which structural and behavioural interoperability standards are unified rather than adjacently deployed. That is a genuine and specific gap.

**Where composition remains unsolved.** Aggregation hierarchies [33], hierarchical ecosystems [4] and twin webs [37] all address composition, and the corpus still reports "a notable absence of architectures that support" it for heterogeneous twins [43]. A community analysis of integration challenges names quality assurance and consistency of composed twins as open problems in every use case it examines, both vertically, between a twin's models and services, and horizontally, where composed twins may command contradictory actuation [106]. The unresolved questions are consistency between overlapping twins of the same asset, arbitration when composed twins disagree, and versioning across a federation whose members evolve independently. Naming schemes [38] and uniform interfaces [42] exist; consistency semantics do not.

**Where the human sits.** Human involvement is architecturally decisive in three of the five case studies, yet it appears mostly as narrative rather than structure. Digital triplet formulations make the human explicit and even measure latency across operating modes [107], and triple human-twin architectures decompose human behaviour data alongside physical, model and service data [108]. The urban mobility federation goes furthest in treating human-in-the-loop control as a design pillar rather than a limitation [89]. But no source in the corpus offers a general treatment of the human as an architectural element with defined interfaces, latency characteristics and failure modes, which is what is needed if human-in-the-loop is to be designed rather than merely tolerated.

**Cross-domain generalisation.** Comprehensive research-gap analyses exist at national-academy level [109], and the corpus supports the claim that twin concepts differ "as the context of application changes (aerospace, manufacturing, city management)" and carry "their own specificity within the life cycle phase of the product" [110]. Adoption barriers have

been catalogued for process industries [111], and platform maturity varies markedly across sectors [74]. What is missing is empirical work testing whether a reference architecture developed in one domain can be instantiated in another. The predictive-maintenance reference architecture comes closest, deriving three application architectures across tractor clutch monitoring, semiconductor polishing and electro-optical health monitoring [18], but these are three applications of one narrow function rather than a genuine cross-domain test.

## 10. Conclusion

Reading the digital twin literature architecturally produces a consistent picture. The field's dominant patterns are layering and service orientation, applied together more often than separately [1]; its dominant quality concern is maintainability under a coupled asset lifecycle [1], [5]; its most portable tactic is the adapter, appearing under many names in every case study examined here; and its most common honest position is the shadow rather than the twin [11].

The five example architectures show what is at stake. A manufacturing platform organises for variability across a product family [80]. A hospital twin organises around an immovable legacy boundary and buys independence with a standardisation tier it expects to become unnecessary [84]. A mobility twin organises by deployment tier on a commercial cloud, pays a measured latency price for it, and keeps that price out of the control loop by leaving control on the vehicle [88]. A wind turbine twin organises as an estimation pipeline under a deliberate sensing budget and states precisely which stages it has not built [92]. A built-asset twin organises around a mandatory internal broker and discovers, at integration time, that abstraction must be applied uniformly across peer subsystems or not at all [97].

None of these is a special case of the others. That is the finding. The value of a digital twin reference architecture, on this evidence, lies not in prescribing a layer stack but in making explicit the decisions that all five had to make and that most publications leave implicit: the integration level and therefore the authority model; the separation criterion that generates the layers; the placement of each component on the cloud-edge continuum and the fidelity that placement permits; the adapter boundaries at which heterogeneity is absorbed; the latency budget for each service class, stated numerically; and the mechanism by which the twin will absorb change in the asset it mirrors. A field that documented those six decisions as consistently as it documents its diagrams would be considerably easier to build in.

## References


[1] E. Ferko, A. Bucaioni, and M. Behnam, "Architecting Digital Twins," *IEEE Access*, vol. 10, pp. 50335–50350, 2022, doi: 10.1109/ACCESS.2022.3172964.

[2] M. Dalibor *et al.*, "A Cross-Domain Systematic Mapping Study on Software Engineering for Digital Twins," *Journal of Systems and Software*, vol. 193, p. 111361, Nov. 2022, doi: 10.1016/j.jss.2022.111361.

[3] V. Villani, M. Picone, M. Mamei, and L. Sabattini, "A Digital Twin Driven Human-Centric Ecosystem for Industry 5.0," *IEEE Transactions on Automation Science and Engineering*, vol. 22, pp. 11291–11303, 2025, doi: 10.1109/TASE.2024.3410703.

[4] M. Martinelli, J. Zhang, A.-K. Splettstoßer, M. Picone, M. Lippi, and A. Wortmann, "Hierarchical Digital Twin Ecosystem for Industrial Manufacturing Scenarios," in *2024 50th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, Aug. 2024, pp. 56–63. doi: 10.1109/SEAA64295.2024.00018.

[5] T. Alskaif *et al.*, "Evolution at the Core of Digital Twin Engineering," Grand Rapids, USA: IEEE, Jul. 2025. doi: 10.5283/epub.77656.

[6] J. Mertens, S. Klikovits, F. Bordeleau, J. Denil, and Ø. Haugen, "Continuous Evolution of Digital Twins using the DarTwin Notation." arXiv, Oct. 2024. Accessed: Nov. 11, 2024. [Online]. Available: http://arxiv.org/abs/2410.23389

[7] S. A. Niederer, M. S. Sacks, M. Girolami, and K. Willcox, "Scaling digital twins from the artisanal to the industrial," *Nature Computational Science*, vol. 1, no. 5, pp. 313–320, May 2021, doi: 10.1038/s43588-021-00072-5.

[8] M. G. Kapteyn, J. V. R. Pretorius, and K. E. Willcox, "A probabilistic graphical model foundation for enabling predictive digital twins at scale," *Nature Computational Science*, vol. 1, no. 5, pp. 337–347, May 2021, doi: 10.1038/s43588-021-00069-0.

[9] A. Thelen *et al.*, "A comprehensive review of digital twin — part 1: Modeling and twinning enabling technologies," *Structural and Multidisciplinary Optimization*, vol. 65, no. 12, p. 354, Nov. 2022, doi: 10.1007/s00158-022-03425-4.

[10] A. Thelen *et al.*, "A comprehensive review of digital twin—part 2: Roles of uncertainty quantification and optimization, a battery digital twin, and perspectives," *Structural and Multidisciplinary Optimization*, vol. 66, no. 1, p. 1, Dec. 2022, doi: 10.1007/s00158-022-03410-x.

[11] W. Kritzinger, M. Karner, G. Traar, J. Henjes, and W. Sihn, "Digital Twin in manufacturing: A categorical literature review and classification," *Ifac-PapersOnline*, vol. 51, no. 11, pp. 1016–1022, 2018, Accessed: Dec. 11, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896318316021

[12] G. Lugaresi and H. Vangheluwe, *From Digital Twins to Twinning Systems*. 2025.

[13] G. Shao, S. Frechette, and V. Srinivasan, "An Analysis of the New ISO 23247 Series of Standards on Digital Twin Framework for Manufacturing," American Society of Mechanical Engineers Digital Collection, Sep. 2023. doi: 10.1115/MSEC2023-101127.

[14] F. Tao *et al.*, "Five-dimension digital twin model and its ten applications," *Comput. Integr. Manuf. Syst*, vol. 25, no. 1, pp. 1–18, 2019.

[15] N. Li *et al.*, "Six-dimensional digital twin modeling and software platform design for complex industrial systems," *Journal of Intelligent Manufacturing*, Jan. 2025, doi: 10.1007/s10845-025-02567-8.

[16] Q. Qi *et al.*, "Enabling technologies and tools for digital twin," *Journal of Manufacturing Systems*, vol. 58, pp. 3–21, Jan. 2021, doi: 10.1016/j.jmsy.2019.10.001.

[17] J. Pfeiffer *et al.*, "Towards a Unifying Reference Model for Digital Twins of Cyber-Physical Systems." arXiv, Jul. 2025. doi: 10.48550/arXiv.2507.04871.

[18] R. van Dinter, B. Tekinerdogan, and C. Catal, "Reference architecture for digital twin-based predictive maintenance systems," *Computers & Industrial Engineering*, vol. 177, p. 109099, Mar. 2023, doi: 10.1016/j.cie.2023.109099.

[19] D. Lehner, J. Pfeiffer, S. Klikovits, A. Wortmann, and M. Wimmer, "A Method for Template-based Architecture Modeling and its Application to Digital Twins," Accessed: Sep. 04, 2024. [Online]. Available: https://www.jot.fm/issues/issue_2024_03/article8.pdf

[20] F. Schnicke, T. Kuhn, and P. O. Antonino, "Enabling Industry 4.0 Service-Oriented Architecture Through Digital Twins," in *Software Architecture*, H. Muccini, P. Avgeriou, B. Buhnova, J. Camara, M. Caporuscio, M. Franzago, A. Koziolek, P. Scandurra, C. Trubiani, D. Weyns, and U. Zdun, Eds., Cham: Springer International Publishing, 2020, pp. 490–503. doi: 10.1007/978-3-030-59155-7_35.

[21] Y. Li and Y. Zhang, "Digital twin for Industrial Internet," *Fundamental Research*, vol. 4, no. 1, pp. 21–24, Jan. 2024, doi: 10.1016/j.fmre.2023.01.005.

[22] D. Zhong, Z. Xia, Y. Zhu, and J. Duan, "Overview of predictive maintenance based on digital twin technology," *Heliyon*, vol. 9, no. 4, p. e14534, Apr. 2023, doi: 10.1016/j.heliyon.2023.e14534.

[23] B. Tekinerdogan and C. Verdouw, "Systems Architecture Design Pattern Catalog for Developing Digital Twins," *Sensors*, vol. 20, no. 18, p. 5103, Jan. 2020, doi: 10.3390/s20185103.

[24] D. Lehner, S. Sint, M. Eisenberg, and M. Wimmer, "A pattern catalog for augmenting Digital Twin models with behavior," *at - Automatisierungstechnik*, vol. 71, no. 6, pp. 423–443, Jun. 2023, doi: 10.1515/auto-2022-0144.

[25] E. Kamburjan, N. Bencomo, S. L. Tapia Tarifa, and E. B. Johnsen, "Declarative Lifecycle Management in Digital Twins," in *Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems*, Linz Austria: ACM, Sep. 2024, pp. 353–363. doi: 10.1145/3652620.3688248.

[26] K. M. Alam and A. El Saddik, "C2PS: A digital twin architecture reference model for the cloud-based cyber-physical systems," *IEEE access*, vol. 5, pp. 2050–2062, 2017, Accessed: Jan. 08, 2025. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7829368/

[27] M. Picone, M. Mamei, and F. Zambonelli, "A Flexible and Modular Architecture for Edge Digital Twin: Implementation and Evaluation," *ACM Transactions on Internet of Things*, vol. 4, no. 1, pp. 8:1–8:32, Feb. 2023, doi: 10.1145/3573206.

[28] H. Xu, J. Wu, Q. Pan, X. Guan, and M. Guizani, "A Survey on Digital Twin for Industrial Internet of Things: Applications, Technologies and Tools," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 4, pp. 2569–2598, 2023, doi: 10.1109/COMST.2023.3297395.

[29] A. Barbone, S. Burattini, M. Martinelli, M. Picone, A. Ricci, and A. Virdis, "Digital Twin Continuum: A Key Enabler for Pervasive Cyber-Physical Environments," in *2024 33rd International Conference on Computer Communications and Networks (ICCCN)*, Jul. 2024, pp. 1–9. doi: 10.1109/ICCCN61486.2024.10637565.

[30] P. Bellavista, N. Bicocchi, M. Fogli, C. Giannelli, M. Mamei, and M. Picone, "Exploiting microservices and serverless for Digital Twins in the cloud-to-edge continuum," *Future Generation Computer Systems*, vol. 157, pp. 275–287, Aug. 2024, doi: 10.1016/j.future.2024.03.052.

[31] A. G. Wermann and J. A. Wickboldt, "KTWIN: A Serverless Kubernetes-based Digital Twin Platform." arXiv, Aug. 2024. doi: 10.48550/arXiv.2408.01635.

[32] M. Vaezi *et al.*, "Digital twins from a networking perspective," *IEEE Internet of Things Journal*, vol. 9, no. 23, pp. 23525–23544, 2022, Accessed: Jan. 08, 2025. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9863238/

[33] C. Human, A. Basson, and K. Kruger, "A design framework for a system of digital twins and services," *Computers in Industry*, vol. 144, p. 103796, Jan. 2023, doi: 10.1016/j.compind.2022.103796.

[34] A. Villalonga *et al.*, "A decision-making framework for dynamic scheduling of cyber-physical production systems based on digital twins," *Annual Reviews in Control*, vol. 51, pp. 357–373, Jan. 2021, doi: 10.1016/j.arcontrol.2021.04.008.

[35] S. Liu, P. Zheng, and B. Jinsong, "Digital Twin-based manufacturing system: A survey based on a novel reference model," *Journal of Intelligent Manufacturing*, vol. 35, pp. 1–30, Jul. 2023, doi: 10.1007/s10845-023-02172-7.

[36] W. Jia, W. Wang, and Z. Zhang, "From simple digital twin to complex digital twin Part I: A novel modeling method for multi-scale and multi-scenario digital twin," *Advanced Engineering Informatics*, vol. 53, p. 101706, Aug. 2022, doi: 10.1016/j.aei.2022.101706.

[37] A. Ricci, A. Croatti, S. Mariani, S. Montagna, and M. Picone, "Web of Digital Twins," *ACM Transactions on Internet Technology*, vol. 22, no. 4, pp. 1–30, Nov. 2022, doi: 10.1145/3507909.

[38] K. Rothermel, O. Herzog, and S. Zhiqiang, *A Proximity Model for Distributed Digital Twin Systems*. 2024. doi: 10.36227/techrxiv.171779090.06097956/v1.

[39] J. Michael, J. Pfeiffer, B. Rumpe, and A. Wortmann, "Integration challenges for digital twin systems-of-systems," in *Proceedings of the 10th IEEE/ACM International Workshop on Software Engineering for Systems-of-Systems and Software Ecosystems*, in SESoS '22. New York, NY, USA: Association for Computing Machinery, Nov. 2022, pp. 9–12. doi: 10.1145/3528229.3529384.

[40] P. Kuruppuarachchi, S. Rea, and A. McGibney, "An architecture for composite digital twin enabling collaborative digital ecosystems," in *2022 IEEE 25th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, IEEE, 2022, pp. 980–985. Accessed: Feb. 25, 2025. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9776073/

[41] L. Heaton, "Platform Stack Architectural Framework: An Introductory Guide."

[42] A. Giulianelli, S. Burattini, A. Ciortea, and A. Ricci, "Engineering Interoperable Ecosystems of Digital Twins: A Web-based Approach," in *Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems*, in MODELS Companion '24. New York, NY, USA: Association for Computing Machinery, Oct. 2024, pp. 476–485. doi: 10.1145/3652620.3688263.

[43] M. Amadeo, C. Marche, G. Ruggeri, S. Ranjbaran, and M. Nitti, "Composing Digital Twins for Internet of Everything Applications: A User-Centric Perspective," in *2024 IEEE International Mediterranean Conference on Communications and Networking (MeditCom)*, Jul. 2024, pp. 73–78. doi: 10.1109/MeditCom61057.2024.10621154.

[44] F. Kazman, S. Echeverria Galvez, and J. Ivers, "A Holistic View of Architecture Definition, Evolution, and Analysis," Carnegie Mellon University, report, Aug. 2023. doi: 10.1184/R1/20722402.v1.

[45] F. Bordeleau, A. Motamedi, and É. Poirier, "A DevOps Approach for the Systematic Development and Evolution of Built Assets Digital Twins," Accessed: Nov. 05, 2024. [Online]. Available: https://itc.scix.net/pdfs/w78-2024-paper_114.pdf

[46] J. Hugues, J. Yankel, J. Hudak, and A. Hristozov, "Twinops: Digital twins meets devops," *CARNEGIE-MELLON UNIV PITTSBURGH PA, Tech. Rep.*, 2022, Accessed: Jan. 08, 2025. [Online]. Available: https://apps.dtic.mil/sti/trecms/pdf/AD1168471.pdf

[47] J. Hugues, A. Hristosov, J. J. Hudak, and J. Yankel, "TwinOps - DevOps meets model-based engineering and digital twins for the engineering of CPS," in *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*, in MODELS '20. New York, NY, USA: Association for Computing Machinery, Oct. 2020, pp. 1–5. doi: 10.1145/3417990.3421446.

[48] D. Lehner, A. Garmendia, and M. Wimmer, "Towards Flexible Evolution of Digital Twins with Fluent APIs," in *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA )*, Sep. 2021, pp. 1–4. doi: 10.1109/ETFA45728.2021.9613222.

[49] G. Beaumont, A. Beugnard, S. Martínez, C. Urtado, and S. Vauttier, "Towards Automating the Life Cycle Management of Digital Twins," in *ER2025-44th International Conference on Conceptual Modeling*, 2025. Accessed: Oct. 22, 2025. [Online]. Available: https://hal.science/hal-05225783/

[50] J. Michael *et al.*, "Model-Driven Engineering for Digital Twins: Opportunities and Challenges," *Systems Engineering*, vol. 28, no. 5, pp. 659–670, 2025, doi: 10.1002/sys.21815.

[51] K. Fiter, L. Malassigné-Onfroy, and B. Oakes, "DTInsight: A Tool for Explicit, Interactive, and Continuous Digital Twin Reporting." arXiv, Aug. 2025. doi: 10.48550/arXiv.2508.18431.

[52] A. Barbie and W. Hasselbring, "Toward Reproducibility of Digital Twin Research: Exemplified with the PiCar-X." arXiv, Aug. 2024. doi: 10.48550/arXiv.2408.13866.

[53] S. Gil, B. Oakes, C. Gomes, M. Frasheri, and P. Larsen, "Toward a systematic reporting framework for Digital Twins: A cooperative robotics case study," *SIMULATION*, Aug. 2024, doi: 10.1177/00375497241261406.

[54] S. Mihai *et al.*, "Digital Twins: A Survey on Enabling Technologies, Challenges, Trends and Future Prospects," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 4, pp. 2255–2291, 2022, doi: 10.1109/COMST.2022.3208773.

[55] A. Hakiri, A. Gokhale, S. B. Yahia, and N. Mellouli, "A comprehensive survey on digital twin for future networks and emerging Internet of Things industry," *Computer Networks*, vol. 244, p. 110350, May 2024, doi: 10.1016/j.comnet.2024.110350.

[56] N. Anwer, R. Stark, F. Tao, and J. Erkoyuncu, "Developing and leveraging digital twins in engineering design," *CIRP Annals*, vol. 2025, Jun. 2025, doi: 10.1016/j.cirp.2025.05.002.

[57] K. Duran *et al.*, "Toward Digital Twin-as-a-Service (DTaaS) Platforms: A Survey on Architecture, Design Requirements, and Performance Metrics," *IEEE Communications Surveys & Tutorials*, vol. 28, pp. 1845–1878, 2026, doi: 10.1109/COMST.2025.3635582.

[58] P. Muñoz, J. Troya, and A. Vallecillo, "Towards Measuring Digital Twins Fidelity at Runtime," in *Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems*, Linz Austria: ACM, Sep. 2024, pp. 507–512. doi: 10.1145/3652620.3688267.

[59] M. Frasheri *et al.*, "Addressing time discrepancy between digital and physical twins," *Robotics and Autonomous Systems*, vol. 161, p. 104347, Mar. 2023, doi: 10.1016/j.robot.2022.104347.

[60] L. Esterle, M. Frasheri, and P. G. Larsen, "Autonomous Reconfiguration Enabled by Digital Twins," in *The Engineering of Digital Twins*, J. Fitzgerald, C. Gomes, and P. G. Larsen, Eds., Cham: Springer International Publishing, 2024, pp. 345–362. doi: 10.1007/978-3-031-66719-0_14.

[61] "Asset Administration Shell Part - 1."

[62] M. Jacoby *et al.*, "Open-Source Implementations of the Reactive Asset Administration Shell: A Survey," *Sensors*, vol. 23, p. 5229, May 2023, doi: 10.3390/s23115229.

[63] J. Zhang, C. Ellwein, M. Heithoff, J. Michael, and A. Wortmann, "Digital twin and the asset administration shell," 2024, Accessed: Apr. 09, 2025. [Online]. Available: https://awortmann.github.io/downloads/paper/Digital_twin_and_the_asset_administration_shell.pdf

[64] J. Mattila, R. Ala-Laurinaho, J. Autiosalo, and K. Tammi, "Interoperability of Digital Twins for Automation With Digital Twin Schema," *IEEE Access*, vol. 13, pp. 200595–200608, 2025, doi: 10.1109/ACCESS.2025.3633736.

[65] C. Schmidt, F. Volz, L. Stojanovic, and G. Sutschet, "Increasing Interoperability between Digital Twin Standards and Specifications: Transformation of DTDL to AAS," *Sensors*, vol. 23, p. 7742, Sep. 2023, doi: 10.3390/s23187742.

[66] S. Cavalieri and S. Gambadoro, "Proposal of Mapping Digital Twins Definition Language to Open Platform Communications Unified Architecture," *Sensors*, vol. 23, p. 2349, Feb. 2023, doi: 10.3390/s23042349.

[67] J. Pfeiffer, D. Lehner, A. Wortmann, and M. Wimmer, "Modeling capabilities of digital twin platforms-old wine in new bottles?" *J. Object Technol.*, vol. 21, no. 3, pp. 3–1, 2022, Accessed: Sep. 04, 2024. [Online]. Available: https://raw.githubusercontent.com/awortmann/awortmann.github.io/master/downloads/paper/Modeling_Capabilities_of_Digital_Twin_Platforms_-_Old_Wine_in_New_Bottles.pdf

[68] A. Budiardjo and D. Migliori, "Digital twin system interoperability framework," Tech. rep. Digital Twin Consortium, East Lansing, Michigan, 2021. Accessed: Feb. 25, 2025. [Online]. Available: https://www.digitaltwinconsortium.org/pdf/Digital-Twin-System-Interoperability-Framework-12072021.pdf

[69] G. Schweiger *et al.*, "An empirical survey on co-simulation: Promising standards, challenges and research needs," *Simulation Modelling Practice and Theory*, vol. 95, pp. 148–163, Sep. 2019, doi: 10.1016/j.simpat.2019.05.001.

[70] P. G. Larsen, J. Fitzgerald, and C. Gomes, "Engineering Digital Twins for Cyber-Physical Systems," in *The Engineering of Digital Twins*, J. Fitzgerald, C. Gomes, and P. G. Larsen, Eds., Cham: Springer International Publishing, 2024, pp. 3–17. doi: 10.1007/978-3-031-66719-0_1.

[71] C. Friedrich, A. Lombana, J. Fasquel, C. Schlick, N. Bennani, and M. Mendil, "CoFMPy: A Python Framework for Rapid Prototyping of FMI-based Digital Twins," in *The 2nd International Conference on Engineering Digital Twins*, 2025. Accessed: Nov. 10, 2025. [Online]. Available: https://hal.science/hal-05326255/

[72] S. Infante *et al.*, "Integrating FMI and ML/AI models on the open-source digital twin framework OpenTwins," *Software Practice and Experience*, Mar. 2024, doi: 10.1002/spe.3322.

[73] S. Gil, P. H. Mikkelsen, C. Gomes, and P. G. Larsen, "Survey on open-source digital twin frameworks–A case study approach," *Software: Practice and Experience*, vol. 54, no. 6, pp. 929–960, Jun. 2024, doi: 10.1002/spe.3305.

[74] A. Khoshkenar and H. Nassereddine, *Exploring Digital Twin platforms across industries*. 2024. doi: 10.22260/ISARC2024/0119.

[75] C. Alcaraz and J. Lopez, "Digital Twin: A Comprehensive Survey of Security Threats," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 3, pp. 1475–1503, 2022, doi: 10.1109/COMST.2022.3171465.

[76] N. Alhamam, M. M. Hafizur Rahman, and A. Aljughaiman, "A Comprehensive Review on Cybersecurity of Digital Twins Issues, Challenges, and Future Research Directions," *IEEE Access*, vol. 13, pp. 45106–45124, 2025, doi: 10.1109/ACCESS.2025.3545004.

[77] T. Kulik, Z. Kazemi, and P. G. Larsen, "Security and Privacy-related Issues in a Digital Twin Context," in *The Engineering of Digital Twins*, J. Fitzgerald, C. Gomes, and P. G. Larsen, Eds., Cham: Springer International Publishing, 2024, pp. 313–344. doi: 10.1007/978-3-031-66719-0_13.

[78] A. R. Qureshi, A. Asensio, M. Imran, J. Garcia, and X. Masip-Bruin, "A survey on security enhancing Digital Twins: Models, applications and tools," *Computer Communications*, vol. 238, p. 108158, Jun. 2025, doi: 10.1016/j.comcom.2025.108158.

[79] S. Rose, O. Borchert, S. Mitchell, and S. Connelly, "Zero Trust Architecture," National Institute of Standards; Technology, Aug. 2020. doi: 10.6028/NIST.SP.800-207.

[80] F. Tao *et al.*, "makeTwin: A reference architecture for digital twin software platform," *Chinese Journal of Aeronautics*, vol. 37, no. 1, pp. 1–18, Jan. 2024, doi: 10.1016/j.cja.2023.05.002.

[81] G. Shao, "Use Case Scenarios for Digital Twin Implementation Based on ISO 23247," *NIST*, May 2021, Accessed: Mar. 06, 2024. [Online]. Available: https://www.nist.gov/publications/use-case-scenarios-digital-twin-implementation-based-iso-23247

[82] G. Shao and M. Helu, "Framework for a digital twin in manufacturing: Scope and requirements," *Manufacturing Letters*, vol. 24, pp. 105–107, Apr. 2020, doi: 10.1016/j.mfglet.2020.04.004.

[83] E. Ferko, A. Bucaioni, P. Pelliccione, and M. Behnam, "Standardisation in Digital Twin Architectures in Manufacturing," in *2023 IEEE 20th International Conference on Software Architecture (ICSA)*, Mar. 2023, pp. 70–81. doi: 10.1109/ICSA56044.2023.00015.

[84] S. Burattini *et al.*, "An Ecosystem of Digital Twins for Operating Room Management," in *2023 IEEE 36th International Symposium on Computer-Based Medical Systems (CBMS)*, Jun. 2023, pp. 770–775. doi: 10.1109/CBMS58004.2023.00317.

[85] R. Sieve, P. Kobialka, L. Slaughter, R. Schlatte, E. B. Johnsen, and S. L. T. Tarifa, "BedreFlyt: Improving Patient Flows through Hospital Wards with Digital Twins," *Electronic Proceedings in Theoretical Computer Science*, vol. 418, pp. 1–15, May 2025, doi: 10.4204/EPTCS.418.1.

[86] Y. Liu *et al.*, "A Novel Cloud-Based Framework for the Elderly Healthcare Services Using Digital Twin," *IEEE Access*, vol. 7, pp. 49088–49101, 2019, doi: 10.1109/ACCESS.2019.2909828.

[87] E. Katsoulakis *et al.*, "Digital twins for health: A scoping review," *npj Digital Medicine*, vol. 7, no. 1, pp. 1–11, Mar. 2024, doi: 10.1038/s41746-024-01073-0.

[88] Z. Wang *et al.*, "Mobility digital twin: Concept, architecture, case study, and future challenges," *IEEE Internet of Things Journal*, vol. 9, no. 18, pp. 17452–17467, 2022, Accessed: Jan. 08, 2025. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9724183/

[89] J. Li, J. Grübel, A. Nadi, M. Snelder, B. van Arem, and J. Gao, "Digital Twin Federation for Urban Mobility Assessment: A Functional Architecture for Low-Car Transformations in the Netherlands," 2025, Accessed: Oct. 25, 2025. [Online]. Available: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=5370630

[90] M. Herath, M. Alvi, R. Minerva, H. Dutta, N. Crespi, and S. M. Raza, "Smart City Digital Twins: A Modular and Adaptive Architecture for Real-Time Data-Driven Urban Management," in *2024 20th International Conference on Network and Service Management (CNSM)*, IEEE, 2024, pp. 1–7. Accessed: Jul. 11, 2025. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10814627/?casa_token=b3lXuilhxVEAAAAA:WzY-AJNhNhitSev00P-EOsrAct0Rzn5kMII2--FmI-MWhs7zhfPeakDTrSPTTaTbbfDKj8c

[91] K. Alexandridis and M. LaFontaine, "Social and Human Dimensions of Digital Twin Technologies in Formal and Informal Institutional Settings," in *Digital Twin: Fundamentals and Applications*, S. Sabri, K. Alexandridis, and N. Lee, Eds., Cham: Springer Nature Switzerland, 2024, pp. 211–236. doi: 10.1007/978-3-031-67778-6_10.

[92] E. Branlard, J. Jonkman, C. Brown, and J. Zhang, "A digital twin solution for floating offshore wind turbines validated using a full-scale prototype," *Wind Energy Science*, vol. 9, no. 1, pp. 1–24, Jan. 2024, doi: 10.5194/wes-9-1-2024.

[93] F. Stadtmann and A. Rasheed, "Diagnostic Digital Twin for Anomaly Detection in Floating Offshore Wind Energy." arXiv, Jun. 2024. doi: 10.48550/arXiv.2406.02775.

[94] P. U. Haselbach *et al.*, "Blade research and demonstration platform," *Journal of Physics: Conference Series*, vol. 1618, no. 5, p. 052073, Sep. 2020, doi: 10.1088/1742-6596/1618/5/052073.

[95] R. van Dinter, B. Tekinerdogan, and C. Catal, "Predictive maintenance using digital twins: A systematic literature review," *Information and Software Technology*, vol. 151, p. 107008, Nov. 2022, doi: 10.1016/j.infsof.2022.107008.

[96] J. Reynolds, S. Sabri, and B. Lee, "Digital Twins for Creating Value Through 'Buildings as Batteries' Using a Mass Customization Network," in *Digital Twin: Fundamentals and Applications*, S. Sabri, K. Alexandridis, and N. Lee, Eds., Cham: Springer Nature Switzerland, 2024, pp. 191–209. doi: 10.1007/978-3-031-67778-6_9.

[97] S. Aissat, J. Beaulieu, É. Poirier, A. Motamedi, J. Gascon-Samson, and F. Bordeleau, "A devops framework for the systematic engineering and evolution of digital twins for built assets," *Software and Systems Modeling*, Oct. 2025, doi: 10.1007/s10270-025-01330-0.

[98] S. Aissat, J. Beaulieu, F. Bordeleau, J. Gascon-Samson, E. A. Poirier, and A. Motamedi, "JuNo-OPS: A DevOps Framework for the Engineering of Digital Twins for Built Assets," in *Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems*, in MODELS Companion '24. New York, NY, USA: Association for Computing Machinery, Oct. 2024, pp. 496–506. doi: 10.1145/3652620.3688266.

[99] P. Zech, C. Nardin, S. Ristov, M. Flora, and R. Breu, "Digital-Twins-as-a-Service in Construction Engineering," in *2024 IEEE 20th International Conference on Automation Science and Engineering (CASE)*, Aug. 2024, pp. 3004–3010. doi: 10.1109/CASE59546.2024.10711409.

[100] S.-W. Yang, Y. Lee, and S.-A. Kim, "Design and Validation of a Real-Time Maintenance Monitoring System Using BIM and Digital Twin Integration," *Buildings*, vol. 15, p. 1312, Apr. 2025, doi: 10.3390/buildings15081312.

[101] P. Bellavista, N. Bicocchi, M. Fogli, C. Giannelli, M. Mamei, and M. Picone, "An Entanglement-Aware Middleware for Digital Twins," *ACM Trans. Internet Things*, Oct. 2024, doi: 10.1145/3699520.

[102] Z. Tang, D. Zhuang, and J. Zhang, "Evaluation framework for domain-specific digital twin platforms," *Scientific Reports*, vol. 15, no. 1, p. 10544, Mar. 2025, doi: 10.1038/s41598-024-82154-8.

[103] P. Talasila, P. H. Mikkelsen, S. Gil, and P. G. Larsen, "Realising Digital Twins," in *The Engineering of Digital Twins*, J. Fitzgerald, C. Gomes, and P. G. Larsen, Eds., Cham: Springer International Publishing, 2024, pp. 225–256. doi: 10.1007/978-3-031-66719-0_11.

[104] E. Kamburjan *et al.*, "GreenhouseDT: An Exemplar for Digital Twins," in *Proceedings of the 19th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, in SEAMS '24. New York, NY, USA: Association for Computing Machinery, Jun. 2024, pp. 175–181. doi: 10.1145/3643915.3644108.

[105] B. J. Oakes *et al.*, "Case Studies in Digital Twins," in *The Engineering of Digital Twins*, J. Fitzgerald, C. Gomes, and P. G. Larsen, Eds., Cham: Springer International Publishing, 2024, pp. 257–310. doi: 10.1007/978-3-031-66719-0_12.

[106] B. Combemale *et al.*, "On the Challenges of Integrating Digital Twins," in *2nd International Conference on Engineering Digital Twins (EDTconf 2025)*, 2025. Accessed: Sep. 29, 2025. [Online]. Available: https://inria.hal.science/hal-05221809/

[107] A. Karnoub, D. Stuber, J. Mertes, M. Klar, and L. Yi, "How to understand the role of human in digital twins: A digital triplet concept," *Digital Engineering*, vol. 6, p. 100052, Sep. 2025, doi: 10.1016/j.dte.2025.100052.

[108] D. Shangguan, L. Chen, C. Su, J. Ding, and C. Liu, "A Triple Human-Digital Twin Architecture for Cyber-Physical Systems," *Computer Modeling in Engineering & Sciences*, vol. 131, no. 3, pp. 1557–1578, 2022, doi: 10.32604/cmes.2022.018979.

[109] *Foundational Research Gaps and Future Directions for Digital Twins*. Washington, D.C.: National Academies Press, 2024. doi: 10.17226/26894.

[110] C. Semeraro, M. Lezoche, H. Panetto, and M. Dassisti, "Digital twin paradigm: A systematic literature review," *Computers in Industry*, vol. 130, p. 103469, Sep. 2021, doi: 10.1016/j.compind.2021.103469.

[111] M. Perno, L. Hvam, and A. Haug, "Implementation of digital twins in the process industry: A systematic literature review of enablers and barriers," *Computers in Industry*, vol. 134, p. 103558, Jan. 2022, doi: 10.1016/j.compind.2021.103558.