Software Quality Journal (2026) 34:56
https://doi.org/10.1007/s11219-026-09793-0

RESEARCH

A catalogue of multi-agent system patterns to design Digital Twins considering Quality attributes

Elena Pretel[1] · José J González-García[1] · Elena Navarro[1] · Víctor López-Jaquero[1] · Schahram Dustdar[2,3]

Received: 27 March 2026 / Accepted: 29 September 2026
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2026

Abstract
Digital twins (DTs) are based on a set of DT properties that are the fundamental characteristics that determine how a DT represents, reflects, memorizes, and interacts with its physical counterpart within a specific usage context. The DT properties are the cornerstone to provide DTs with higher quality. Despite the increasing adoption of Multi-Agent Systems (MAS) in DT development, there is a lack of structured guidance on how agent design can support the development of these DT properties. This paper addresses this gap by proposing a catalogue of agent design patterns tailored to support the reflection, memorization and entanglement properties of a DT. Each pattern is aligned with established design patterns in MAS literature. Decision models are also introduced to guide practitioners in selecting appropriate patterns based on the essential aspects of each property. The proposed patterns and design models were applied in a real-world case study involving the design of a DT to manage study sessions for children with ADHD, illustrating their applicability in a case study. The results support the feasibility and utility of our catalogue of design patterns for the development of the selected DT properties that are relevant to provide DT with quality characteristics.

Keywords Digital Twin · Multi-Agent Systems · MAS · Design Patterns

1 Introduction

A DT is commonly defined as a dynamic digital representation of a physical entity that enables real-time monitoring, simulation, prediction, and control of physical assets (Minerva et al., 2020). Since its introduction by Grieves and Vickers (2017) in 2003, the DT

Elena Pretel, José J González-García, Víctor López-Jaquero and Schahram Dustdar these authors contributed equally to this work.

This work is an extension of the article "E. Pretel et al. (2026) An Agent Design Pattern Catalogue for the Entanglement of Digital Twins. In: 18th International Conference on the Quality of Information and Communications Technology. https://doi.org/10.1007/978-3-032-20631-2_1" considered for publication in the Special Issue entitled "Software Quality in an AI-Driven World".

Extended author information available on the last page of the article

Published online: 08 October 2026

concept has evolved significantly, driven by advances in both sensing technologies and data management. Moreover, the integration of AI in the DT has further expanded its capabilities, enabling data-driven modelling, predictive analytics, and autonomous decision-making (Kreuzer et al., 2024). As a result, DTs have become key enablers of intelligent and adaptive systems across domains such as industrial production, healthcare, smart cities, transportation, and energy management (Guo & Lv, 2022). For instance, a DT of a wind turbine can continuously analyse operational data and use AI-based models to predict system behaviour and support automated control actions to optimize energy production or prevent failure (Pretel et al., 2025).

The effectiveness and reliability of the digital services built on top of a DT depend heavily on the quality of its underlying model (Zhang et al., 2022). Quality, in this context, goes beyond computational precision. It involves ensuring that the DT provides an accurate, consistent, and meaningful representation of its physical counterpart (Zhang et al., 2022). High-quality DTs enable dependable monitoring, predictive capabilities, and decision support, which are essential for achieving trustworthy and adaptive cyber-physical systems (Lu et al., 2024). Consequently, developing high-quality DTs has become a central research goal. Researchers have thus focused on constructing increasingly rich and comprehensive digital representations to minimise the gap between physical and digital parts (Zhang et al., 2022).

However, achieving a high-quality DT is not simply a matter of building more detailed models or adding computational complexity (Lu et al., 2024). True quality emerges from the DT's ability to remain semantically meaningful, contextually appropriate, and operationally actionable within its intended environment. In this regard, the properties of a DT defined by Minerva et al. (2020)–*representativeness and contextualization*, *reflection*, *memorization*, and *entanglement*–provide a conceptual foundation that some methodologies, such as MAS4DT (Pretel et al., 2025), use to systematically guide the design of DTs. Minerva et al. (2020) define these properties as:

- *Representativeness and contextualization* refers to the ability of the DT to capture the relevant attributes, relationships, and behaviours of the physical entity in its specific context of use.
- *Reflection* describes the capability of the DT to accurately mirror the state and evolution of the physical entity over time, maintaining a consistent and timely correspondence between physical and digital data.
- *Memorization* refers to the DT ability to maintain a persistent record of the physical twin's past and present, including states, events, and relevant contextual information.
- *Entanglement* defines the degree and direction of coupling between the DT and its physical counterpart, ensuring that both entities can exchange data, stay synchronised, and, in advanced cases, influence each other through bidirectional control mechanisms.

Beyond their relevance to conceptual design, these properties have direct implications for the development of DT, as they drive architectural decisions, such as which communication mechanisms and data management strategies are required to implement a DT in practice. Several architectural paradigms have been proposed to facilitate the development of DT, including IoT-based, microservices, event-driven and cyber-physical architectures (Raghunandan et al., 2023; Xu et al., 2023). Each of them offers useful capabilities for specific concerns, such as device connectivity, interoperability, modular deployment, or tight inte-

gration with the physical environment. However, these approaches do not always provide an explicit mechanism to capture the autonomy, coordination, and goal-directed behaviour that may be required in more adaptive DTs (Hribernik et al., 2021). For this reason, the choice of architecture should be guided not only by communication and data management requirements, but also by the degree of reasoning, autonomy, interaction, and dynamism expected from the DT.

In this context, the use of Multi-Agent Systems (MAS) has emerged as a promising architectural approach to address the complexity of DT development as its core properties are closely aligned with several DT needs (Pretel et al., 2024). MAS explicitly encapsulate autonomous decision-making, coordination, and adaptive behaviour within software agents. This distinction becomes particularly valuable when DTs must continuously interpret contextual information, coordinate distributed decisions, and dynamically adapt their behaviour as the physical environment evolves. MAS consists of multiple autonomous agents that interact with each other and their environment to achieve system-level goals (Dorri et al., 2018). By leveraging agent-oriented paradigm, DTs can go beyond passive mirroring by incorporating reasoning, learning, and goal-directed behaviours. Thus, MAS properties can naturally support the development of DT properties (Pretel et al., 2024). For instance, for the *reflection* property, MAS provide autonomous and reactive agents capable of processing sensor information, and deciding how data should be transformed or fused before updating the status of the DT, thereby improving the consistency and semantic quality of the reflected state. Then, for the *memorization* property, the distributed knowledge management capabilities of MAS enable the DT to maintain, organise, and exploit both current and historical information according to its objectives, supporting efficient reasoning over persistent knowledge rather than simple data storage. Finally, for the *entanglement* property, MAS naturally support the continuous interaction between the digital and physical worlds through agents that perceive data from the physical environment and agents capable of autonomously modifying the behaviour of the physical twin. Agent interaction mechanisms, together with properties such as data transmission frequency and communication protocols facilitate the bidirectional interaction between the physical and digital twins, enabling stronger and more flexible coupling than architectures focused solely on message exchange.

Despite this theoretical alignment, prior research on the synergies between MAS and DTs shows that these remain underexploited in current DT proposals and suggests the need for more systematic ways of translating MAS capabilities into DT architectures (Pretel et al., 2024). Existing proposals, such as the MAS4DT methodology (Pretel et al., 2025), primarily focus on early phases such as requirements analysis and high-level architectural design, and therefore do not yet provide end-to-end guidance for detailed implementation. Furthermore, the interactions between *representativeness and contextualization*, *reflection*, *memorization*, and *entanglement* are seldom examined holistically. Design choices intended to strengthen one property (e.g., increasing entanglement through real-time updates) may unintentionally compromise others (e.g., raising data processing costs and affecting reflection consistency). This complexity underscores the need for architectural principles and reusable patterns to help developers achieve a balance among the essential DT properties.

To address these challenges, this work explores how agent design patterns–recurring, generalised solutions to common design problems in MAS (Wedyan & Abufakher, 2020)–can support the systematic development of the *reflection*, *memorization*, and *entanglement* DT properties. It is important to note that the *representativeness and contextualization* prop-

erty does not have associated design patterns, as it mainly belongs to the initial requirements phase. Nevertheless, it is considered a fundamental quality property, as ensuring a representative and contextually grounded model is essential to build a meaningful DT, and it is explicitly addressed within the MAS4DT methodology (Pretel et al., 2025). This work aims to answer the following research questions:

- *RQ1*: Which agent design patterns can be used to support the development of the *reflection*, *memorization*, and *entanglement* DT properties?
- *RQ2*: How can software architects be guided in selecting and combining these design patterns to achieve the desired balance among these three DT properties?

The main contributions of this paper are twofold. First, to address RQ1, we propose a catalogue of MAS-based design patterns that support the development of the reflection, memorization, and entanglement DT properties. Second, to address RQ2, we introduce decision models that guide software architects in selecting and combining the proposed patterns according to the requirements and characteristics associated with each DT property. Lastly, the applicability of both contributions is illustrated through a running example involving the design of a DT for managing study sessions for children with ADHD.

The remainder of this paper is organised as follows. Section 2 and Section 3 introduce some background about DT and discuss relevant related work, respectively. Then, Section 4 explains the research methods applied in this study. Next, Section 5 shows the running example used to illustrate the design patterns and Section 6 presents each agent design pattern of the catalogue in detail and the decision models for pattern selection. Lastly, Section 7 describes the threats to validity, and Section 8 summarises the conclusions.

2 Background

In its most widely accepted definition, a DT consists of three tightly related components: (i) a physical entity, (ii) a virtual entity, and (iii) the connection between both (Grieves & Vickers, 2017). It is important to distinguish DTs from other related notions. Unlike *digital models*, which provide static and typically offline representations, and *digital shadows*, which enable unidirectional data flows from the physical asset to its virtual counterpart, a DT maintains a bidirectional link (Kritzinger et al., 2018). This bidirectionality supports not only monitoring and analysis but also the execution of control actions from the DT to the physical twin.

As DTs assume increasingly critical roles in decision-making and automation processes, ensuring their quality becomes essential. A high-quality DT is defined by its ability to provide accurate, actionable, and trustworthy digital services, which depends on how effectively the virtual entity remains semantically meaningful and dynamically synchronised with its physical counterpart (Lu et al., 2024). We adopt the properties defined by Minerva et al. (2020)–*representativeness and contextualization*, *reflection*, *memorization*, and *entanglement*–as the essential properties that guarantee DT quality since together they ensure that the DT is properly contextualised, faithfully updated, historically grounded, and reliably coupled to its physical counterpart. The following subsections present each property in detail.

2.1 Representativeness and contextualization DT property

Representativeness and contextualization refers to the capability of a DT to model its physical twin (PT) by selecting the features, that is attributes, relationships, and behaviours that are relevant according to a specific usage context (Minerva et al., 2020). Therefore, the DT does not attempt to replicate all the attributes of the PT but instead focuses on those that are meaningful to its intended objective. The challenge lies in determining which attributes constitute a “good enough" representation without over-engineering the model or compromising its utility across diverse usage scenarios.

Addressing this property is considered in this work as a prerequisite to apply the design patterns here presented. This property is explicitly addressed within the MAS4DT methodology (Pretel et al., 2025) and it includes eliciting, justifying, and validating the selected attributes according to stakeholder needs and application goals. Once this representation has been established, the architectural design phase focuses on how these selected attributes should be reflected, memorised, and synchronised within the DT. Consequently, the design patterns proposed in this paper assume that the DT representation has already been defined during requirements stage and provide guidance for its architectural realization. In this sense, the *representativeness and contextualization* property does not disappear from the design process, rather, the functional requirements associated with each identified attribute constrain and guide the selection of the reflection, memorization, and entanglement patterns presented in this work.

2.2 Reflection DT property

Reflection refers to the DT ability to reflect the current status and meaningful features of its physical twin by incorporating relevant measurements, events, and state information (Minerva et al., 2020). Formally, as illustrated in Fig. 1, if the state of the PT is characterised by a multidimensional vector of observable variables $X = \{x_1, x_2, \ldots, x_n\}$, reflection requires the existence of well-defined transformation functions $f : X \rightarrow X'$ that map physical measurements to congruent digital representations, where congruence may involve different transformations such as normalisation, aggregation, or discretisation (Minerva et al., 2020).

Sensor data from the physical twin can be reflected in the DT through two distinct approaches (Minerva et al., 2020): *direct mappings*, where raw measurements are stored without modification, or *derived data*, where sensor readings undergo preprocessing to generate semantically enriched representations suitable for analysis and decision-making. Below some transformations are described that can be applied to produce *derived data*:

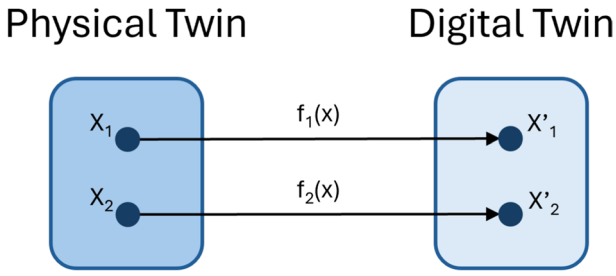


Fig. 1 Relationship between the PT and the DT by means of functional transformations (adapted from Minerva et al. (2020))

- *Analytical attributes* are derived from physics-based or mathematical models. For instance, heat transfer coefficients may be computed from thermodynamic equations. These attributes rely on deterministic models grounded in engineering principles.
- *Statistical attributes* result from data aggregation. Examples include mean values, standard deviations, correlation coefficients, or trend gradients. These metrics summarise large volumes of sensor data.
- *Discrete events* represent instantaneous occurrences or state transitions that are reflected through transformation functions that map raw data from the physical twin to semantically enriched event representations in the DT. For example, transforming a voltage value into 'on' or 'off'.
- *Fusion attributes* are computed by combining multiple heterogeneous sensors. For example, the health status of a machine can be derived from the fusion of vibration and temperature sensors. These attributes provide more reliable information from sensor sets.

2.3 Memorization DT property

Memorization is the ability of the DT property to maintain a persistent storage of the physical twin's current and historical states, events, and contextual information throughout its lifecycle (Minerva et al., 2020). This property is fundamental as it provides the essential historical data required to support other value-added DT properties defined by Minerva et al. (2020), such as *predictability*, which enables the DT to simulate and forecast the physical twin's future behaviour based on accumulated past data.

For the practical implementation of this property, industry standards, such as the *Asset Administration Shell* (AAS) promoted by the Industrial Digital Twin Association (IDTA), offer structured frameworks for lifecycle-long asset representation (IDTA, 2023). The AAS establishes a standardised, machine-readable digital model of physical assets that comprehensively captures their technical features, operational properties, behavioural states, and functional capabilities (IDTA, 2023). Within this standard, attributes of the DT are classified into two categories:

- *Static or constant attributes* are properties that remain invariant over time. They describe the fundamental features of the physical twin and typically change only when a redesign, reconfiguration, or structural modification occurs.
  - *Structural attributes* define the physical nature of the physical twin. This includes aspects such as geometry, material composition, and topological relationships. These attributes determine mechanical behaviour, load distribution, and physical constraints.
  - *Configurational attributes* refer to setup parameters and calibration settings that define how the physical twin operates under normal conditions. These include controller parameters such as operational thresholds, and predefined limits. Although they may occasionally be adjusted, they are not continuously time-varying during operation and therefore remain static for most cases.

  - *Semantic attributes* capture the domain, conceptual and relational structure of the physical twin and its domain. These include logical relationships such as "component-of", "depends-on", or "compatible-with".

- *Dynamic or variable attributes* are time-dependent features that evolve as the DT operates. They represent the behaviour of the system, condition, and performance over time, and are typically obtained through sensors, monitoring systems, or operational logs.

  - The *instantaneous state* reflects the state of the DT at a specific moment. Examples include position, speed, temperature or pressure. These attributes describe the real-time operational snapshot of the system.
  - *Flows or streams* consist of continuous time-series data representing signals collected over time, such as power consumption or acoustic emissions. Unlike isolated instantaneous values, streams allow for frequency-domain analysis, trend detection, and anomaly identification.

2.4 Entanglement DT property

Entanglement refers to the relationship between physical twin and DT. This relationship implies a constant, and as far as possible real-time flow of information, whereby the DT accurately reflects the state of the physical twin (Minerva et al., 2020). This property is recognized as one of the most important aspects in several standardization approaches, including ISO 23247 (ISO, 2021), which provides a general framework for the implementation of DT in industrial manufacturing environments. The standard highlights the importance of maintaining efficient synchronization between physical and digital entities for effective control and monitoring. The entanglement property is based on three aspects (Minerva et al., 2020):

- *Connectivity* (C): there should be a direct or indirect means of communicating state changes and data between the physical and digital twin. Transmission can be direct if the physical twin can communicate directly with its DT, or indirect if it relies on a third entity to send the information.
- *Association* (A): it describes the direction of data exchange between the physical twin and its DT. This association can be unidirectional, for example, sensors that only transmit data to the DT, or bidirectional, when the DT can also act on the physical twin through actuators in order to modify its behaviour.
- *Promptness* (P): data updates must occur at times compatible with the intended use of the DT. In critical applications, such as medical or industrial systems, continuous data flow is required; however, in other cases, periodical or event-driven updates may be sufficient.

Based on these three aspects, the degree of entanglement can vary (Minerva et al., 2020):

- *Weak entanglement* means that data from the physical twin are inferred indirectly by analysing data from the surrounding environment (C). For example, through observation of the environment or analysis of external sources, such as social networks.

- *Simple entanglement* means that there is a unidirectional link (A) between the physical twin and its DT, allowing partial synchronization.
- *Strong entanglement* means that the physical twin is continuously linked to the DT (P) and, moreover, the link is bidirectional (A), so that the DT also has the ability to modify or update the status of the physical twin.

3 Related work

Design patterns are a crucial part of software engineering, especially in the context of system architecture and the development of complex systems. Design patterns are typical solutions to common problems in software design (Wedyan & Abufakher, 2020). Different works have focused on the development of design patterns. In software engineering, the proposal of Gamma et al. (1995), introduced 23 patterns that have become the basis of object-oriented design. These patterns address common problems in software design and have been widely adopted by practitioners. Since then, some extensions and adaptations of these patterns have been proposed to address new emerging challenges, such as the design of patterns for distributed systems (Schmidt et al., 2013) and real-time systems (Douglass, 2003).

Consequently, with the increasing use of MAS for the development of software systems, several works have focused on the development of design patterns for agent-based systems. For instance, Oluyomi et al. (2004) discuss the different abstraction levels and implementation complexities that arise in MAS engineering, showing that the internal structure and interaction logic of agents require design guidance beyond traditional object-oriented solutions. Similarly, Marks et al. (2018) review agent design patterns for assistive systems in multiple domains, illustrating how recurring coordination and reasoning problems can be addressed through reusable agent-oriented solutions. In industrial settings, Luder et al. (2017) introduce design patterns for agent-based production control systems, highlighting the importance of modularity, coordination, and scalability in cyber-physical environments. More recently, Liu et al. (2025) propose a collection of patterns for agents-based on foundational models, which reflects the continued interest in structuring agent capabilities through reusable architectural solutions.

In the field of DTs some architectural and conceptual contributions have been proposed so far. One of the most influential conceptual studies is the survey by Minerva et al. (2020), which identifies the fundamental properties, architectural building blocks, and enabling technologies a DT should have. Rather than prescribing implementation solutions, this work establishes the conceptual foundations of DTs and discusses DT properties such as *representativeness and contextualization*, *reflection*, *entanglement*, and *memorization*. These concepts have recently received increasing attention during the requirements engineering stage of DT development. A recent systematic literature review on requirements engineering for DTs by Macías et al. (2026) identifies DT model construction, exchange of information and data management as recurrent cross-domain requirements, highlighting that they should be addressed from the early stages of DT specification. However, this review focuses on identifying and classifying requirements

rather than providing architectural guidance on how these requirements should be realised through software design decisions. Furthermore, Korenhof et al. (2021) argues that a digital representation should not be understood as a neutral mirror of the physical entity, but rather as the result of modelling decisions regarding system boundaries, abstraction level, selected variables, data sources, and intended objectives. Similarly, the concept of Memory Twins extends the traditional notion of memorization beyond historical data storage by incorporating semantic, historical, and contextual knowledge (Cassar et al., 2025). These studies reinforce the importance of the contribution of the *representativeness and contextualization*, *reflection*, *entanglement*, and *memorization* DT properties for DT quality. Nevertheless, they do not discuss how these concerns can be systematically translated into design patterns.

On the other hand, some works have proposed general methodologies and reference models for DT engineering. For example, the Five-Dimension Digital Twin Model provides a high-level framework composed of the physical entity, the virtual entity, data, services, and connections, serving as a reference architecture for DT development (Tao et al., 2018). Likewise, MAS4DT proposes the development of DTs by using a MAS architecture, identifying the agents needed to achieve the objectives of the DT (Pretel et al., 2025). These approaches provide valuable guidance for conceptualising DT architectures and identifying their main components. However, they remain intentionally technology-agnostic and do not define reusable software design patterns that support the implementation of specific DT properties.

More implementation-oriented contributions have focused on recurring architectural solutions for DTs. Tekinerdogan and Verdouw (2020) propose a catalogue of DT architectural design patterns addressing recurring structural concerns, such as monitoring and control. Their work focuses on the overall architecture of DT applications and the functional role of the DT within a system. More recently, Khail and Schnizer (2024) introduce a set of patterns that support the evolution of DTs from weak to strong entanglement, emphasising synchronisation mechanisms and the degree of coupling between physical and digital counterparts. These patterns specifically target the entanglement property of DTs, but they do not consider other fundamental properties such as reflection and memorization, nor do they address how the entanglement should be implemented from a MAS perspective. On the other hand, Pretel et al. (2024) analyses the synergies between MAS and DT properties, however, this work remains at the level of conceptual correspondences between both paradigms and does not formalise reusable agent-oriented design patterns for implementing DT properties.

In light of the works reviewed above some research has already addressed the DT properties from a conceptual view, the structure of the DT architecture and the DT requirements specification as Table 1 summarizes. However, none of them provides an implementation-oriented MAS pattern catalogue that bridges the gap between early DT specification activities and the detailed realisation of reflection, memorization, and entanglement. This paper addresses this gap by proposing a catalogue of agent design patterns intended to guide the development of some essential DT properties from a MAS perspective.

Table 1 Comparison of related work and the proposed approach

Work	DT properties	DT requirements	DT architecture	MAS architecture	Design patterns	Main difference with this work
Minerva et al. (2020)	Yes	No	Yes	Partially	Partially	Conceptually defines DT properties, but does not provide implementation guidance.
Macías et al. (2026)	Yes	Yes	No	No	No	Identifies and classifies DT requirements, but does not translate them into design solutions.
Korenhof et al. (2021)	Partially	No	No	No	No	Discusses modelling and representation decisions, but not their implementation through patterns.
Cassar et al. (2025)	Partially	No	No	No	No	Broadens the notion of DT memory, but does not propose patterns for its implementation.
Tao et al. (2018)	No	No	Yes	No	No	Provides a high-level DT reference model without reusable implementation patterns.
Pretel et al. (2025)	Yes	Yes	Yes	Yes	No	Guides early MAS-based DT specification and architecture, but not detailed pattern-based implementation.
Tekinerdogan and Verdouw (2020)	Partially	No	Yes	No	Yes	Provides DT architectural patterns, but they are not MAS-specific or explicitly property-driven.
Khail and Schnizer (2024)	Partially	No	Yes	No	Yes	Provides patterns for entanglement property only.
Pretel et al. (2024)	Yes	No	No	Yes	No	Analyses conceptual MAS–DT synergies, but does not formalise design patterns.
This work	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Provides MAS-specific patterns and decision models for reflection, memorization, and entanglement, linking functional requirements with ISO/IEC 25010 non-functional consequences.**

4 Methodology

Figure 2 shows the research methodology conducted in this work, which combines two research methods: Systematic Literature Review (SLR) and Grounded Theory (GT). By integrating these two methods, recurrent structures and agent behaviours related to the reflection, memorization and entanglement properties of DTs were explored.

First, this paper uses the Systematic Literature Review (SLR) on DT and MAS conducted by Pretel et al. (2024). This SLR follows the guidelines proposed by Kitchenham et al. (2009), therefore, the authors focused on the available primary research papers and selected relevant papers based on a set of inclusion and exclusion criteria. Moreover, they conducted a quality

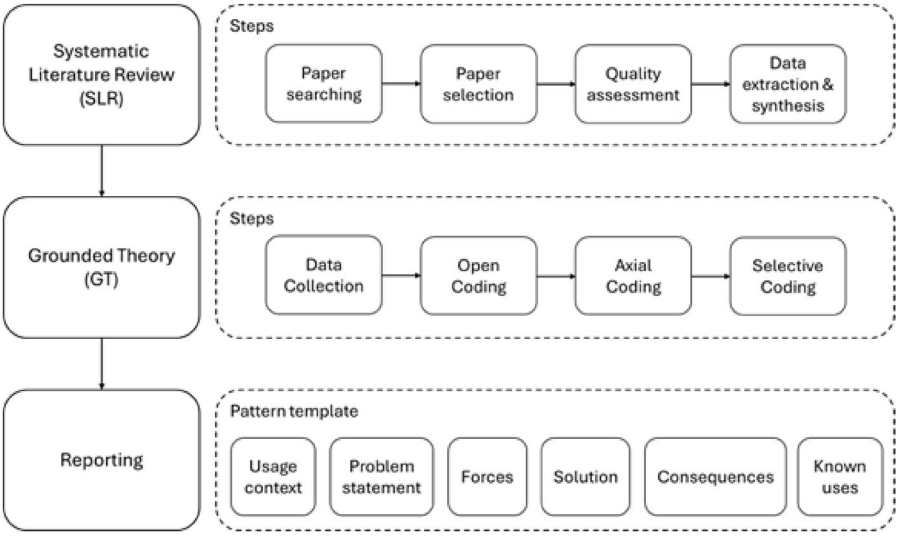


Fig. 2 Methodology (adapted from Liu et al. (2025))

assessment of the selected materials to ensure the quality of the papers. Finally, in this SLR, 64 studies were included for data extraction and synthesis. These studies were first examined by distinguishing between two main groups: (i) MAS with DT: studies in which the DT is developed using a MAS architecture, and (ii) MAS for DT: studies in which MAS agents consume or exploit DT data without the DT itself being implemented as a MAS. Since the purpose of this work is to identify MAS design patterns for implementing DT properties, only the first group was considered for the pattern derivation process. This filtering resulted in a subset of 21 studies, which were further analysed to identify recurring architectural solutions and similar ways of addressing the DT properties considered in this paper. Not all studies provided evidence for all three properties (reflection, memorization and entanglement), but each of them contributed to the identification of one or more pattern candidates. The extracted information from these primary studies was used as the initial input for the pattern identification process.

Then, in order to further enrich the information gathered to define patterns, in this article we extended our research approach by including Grounded Theory (GT) (Gorodetsky et al., 2019). This combination allowed us to complement the SLR results with additional findings derived from a process of inductive and iterative analysis. Firstly, we conducted additional searches focusing on MAS design pattern works that were not captured in the SLR (Pretel et al., 2024). As a first step, an additional search was conducted on Scopus combining the terms MAS, DT, and design patterns in titles, abstracts, and keywords. This search did not return any studies. Therefore, the search scope was broadened by looking for design patterns in all fields while keeping MAS and DT as core terms, which resulted in 13 studies. These studies were also analysed to identify which DT properties were addressed and which MAS patterns were associated with each property. In parallel, an additional search focused on MAS and design patterns in title, abstract, and keywords was conducted. From the resulting body of literature, the most cited studies were examined progressively until 20 papers had been analysed, since at that point no substantially new patterns were being identified and the same patterns began to recur.

Next, open coding was applied to the collected studies to identify design patterns related to *reflection*, *memorization* and *entanglement* DT properties, followed by axial coding to establish connections between these patterns and different aspects of each property (Section 2). During the open coding phase, the studies analysed were reviewed by the first author of the paper, who examined agent roles, coordination mechanisms, and functionalities related to the DT properties. The preliminary coding results were then discussed with the rest of the authors until consensus was reached. In the axial coding phase, the identified MAS design patterns were grouped according to the DT property they supported. This step made it possible to establish synergies between DT and MAS properties (Pretel et al., 2024), so that patterns originally discussed from a MAS perspective could be interpreted in terms of their contribution to DT properties.

Finally, selective coding was applied to create a catalogue of design patterns related to *reflection*, *memorization* and *entanglement* DT properties. In this final phase, the patterns proposed in Section 6 constitute a new set of DT-oriented patterns derived from recurring agent-oriented design solutions identified in the MAS literature.

The reporting of the patterns has been carried out by using the extended template of Meszaros (1997). It includes the name of the pattern, a brief summary, the context of use, a problem statement, the forces, the solution, its consequences, the related design patterns, and some examples of known uses of the pattern in the real world. The patterns in this catalogue are new DT-oriented patterns inspired by recurrent mechanisms and solution structures identified in the MAS design pattern literature analysed, which were reinterpreted and reformulated to address the *reflection*, *memorization*, and *entanglement* properties of the DT.

This research methodology, combining SLR and GT, provided a richer understanding of the design principles and architectural patterns used to develop the *reflection*, *memorization*, and *entanglement* properties. The methodological role of the SLR was to define the relevant corpus of studies on DTs developed by using MAS, whereas GT supported the inductive analysis of MAS design patterns to identify recurring agent-based mechanisms associated with these DT properties. By combining the results of both research methods, we ensured that the identified patterns are based on empirical findings and, at the same time, are theoretically grounded. This analysis provides the basis for the next sections of this article, in which the derived design patterns are presented. To improve reproducibility, the material generated during the analysis can be made available through Zenodo upon request.

5 Running example: ADHD study session management

This Section presents the running example used to illustrate when to apply the design patterns described later in Section 6: a DT for managing study sessions of children diagnosed with Attention Deficit Hyperactivity Disorder (ADHD) (González-García et al., 2025). ADHD is a neurodevelopmental disorder characterised by persistent patterns of inattention, hyperactivity, and impulsivity that significantly interfere with daily functioning and academic performance (Loh et al., 2022). Children with ADHD face substantial challenges in traditional learning environments, including difficulties in maintaining focus during extended periods, organising tasks, following multi-step instructions, and controlling impulsive behaviours that disrupt study routines.

To address those challenges, we developed a DT of a child by using a MAS that continuously monitors the child's physiological and environmental state to personalise and optimise study sessions in real time (see Fig. 3). Therefore, the *physical twin* comprises the child and the study environment, including the wearable device and the environmental sensors and actuators. The *digital twin* developed is a software system that represents, monitors, and supports the management of the study session of the child and their environment conditions. The DT is implemented as a MAS whose agents collectively acquire and process data from the physical twin, maintain the corresponding digital representation, store current and historical information, reason about the conditions of the study session, and coordinate actions or notifications addressed to the wearable device, environmental actuators, and tutor. Thus, the MAS constitutes the underlying software architecture used to implement the DT. The adoption of a MAS architecture for DT implementation was motivated by the Systematic Literature Review on the synergies between DTs and MAS (Pretel et al., 2024), where the potential of using MAS for the development of DTs was analysed.

We used the MAS4DT methodology (Pretel et al., 2025) to identify the DT objectives and the most relevant attributes required to represent the child in the study context (*representativeness and contextualization* DT property). The objectives of the DT are: (1) personalised session adaptation, which dynamically adjusts session duration to match attention capacity; (2) post-session analytics, providing tutors with detailed insights into performance and attention patterns; and (3) optimal environment control, ensuring that external factors such as lighting, noise, and temperature are continuously regulated to maintain ideal learning conditions.

The identified attributes are grouped into static and dynamic categories. Static attributes include the *child's age* (which determines the reference values for the duration of the session) and *interaction restrictions* (limited to a maximum of 2 messages per minute to avoid overwhelming the child). Dynamic attributes comprise physiological and contextual variables that evolve over time and must therefore be continuously updated in the corresponding

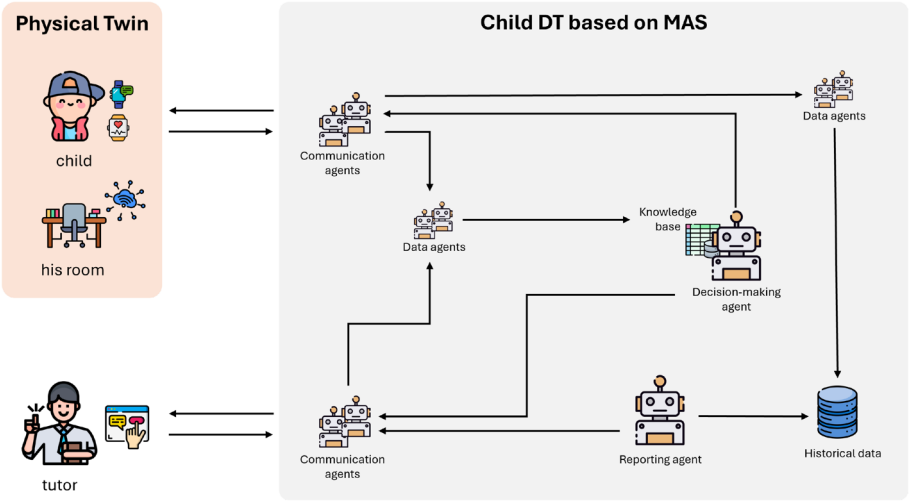


Fig. 3 Child DT based on MAS for managing study sessions

DT status, such as *heart rate*, *level of movement* (derived from accelerometer data to classify restlessness as low, medium, or high), and environmental conditions, including *lighting*, *temperature*, and *noise*. Together, these attributes provide a comprehensive representation of the attention state of the child and the suitability of the environment.

The DT operates within a smart study room instrumented with environmental sensors and controllable actuators, while the child wears a smart wristband that continuously captures physiological and movement data. These data streams are collected in real time, enabling the DT to have an up-to-date representation of the child and environmental state. Based on this continuous monitoring and the custom rules defined by the tutor of each child, the DT can autonomously trigger adaptive interventions, such as shortening or extending the duration of the session, recommending breaks, or dynamically adjusting environmental conditions.

This running example serves as a concrete reference throughout the article to illustrate how the proposed design patterns and decision models can be applied to the development of a DT based on MAS. So, it will be used to better illustrate the application of the patterns and the correspondence between the theoretical concepts and the practical implementation using the JaCaMo MAS framework (JaCaMo, 2022). This MAS framework is integrated by three different dimensions (Fig. 4): i) Jason, which is responsible for managing the agents' reasoning and behaviour through the Belief-Desire-Intention (BDI) paradigm; ii) CArtAgO that provides artifacts to enable the interaction with the environment the agents are embedded in; and, iii) Moise that defines the organizational and normative structures for coordinating agent collaboration. In this DT, Jason was used for the implementation of the agents, while CArtAgO artifacts have been used to facilitate IoT communication with the physical twin. For simplicity, the organizational aspects that would typically be handled by Moise are implemented directly through Jason. The implementation is available upon request.

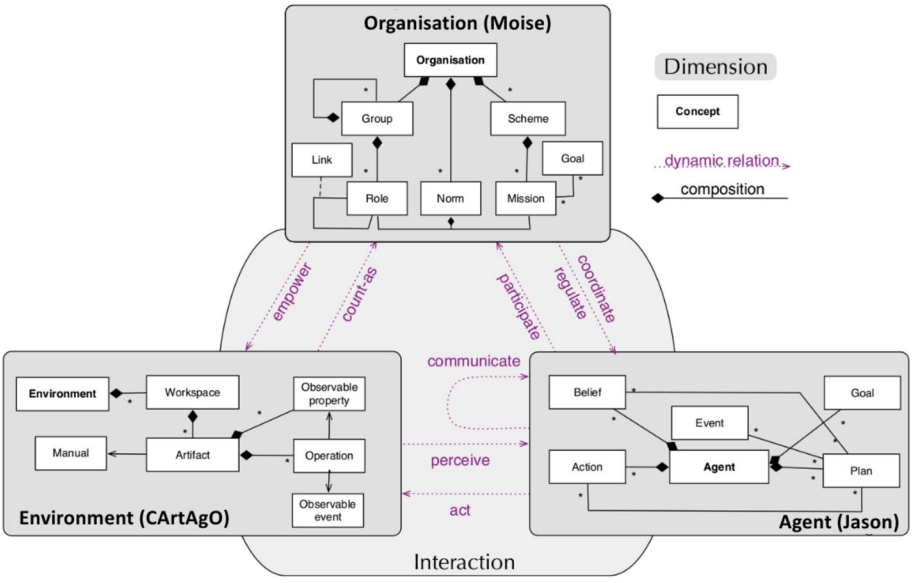


Fig. 4 Architectural overview of the JaCaMo framework, showing the integration between Jason agents, CArtAgO artifacts, and the Moise organization layer (adapted from JaCaMo (2022))

Figure 4 provides an overview of the conceptual organization of JaCaMo MAS approach, showing how its three dimensions, namely Organisation (Moise), Environment (CArtAgO), and Agent (Jason), work together within a common interaction space. As can be observed, JaCaMo relies on the coordinated interplay between organisational structures, environmental resources, and agent reasoning elements. The dynamic relations represented in this Figure refer to runtime interaction dependencies among these dimensions, such as communication, coordination, regulation, participation, perception, and action.

6 Agent Design Pattern Catalogue for essential DT properties

To address the RQ1, this Section presents a catalogue of agent design patterns aimed at supporting the development of *reflection*, *memorization*, and *entanglement* properties of a DT. Each pattern is systematically described by using the format presented in Meszaros (1997). The terms used to describe the *forces*, *benefits* and *drawbacks* of each pattern correspond to quality characteristics or subcharacteristics defined in the ISO/IEC 25010 standard (25010, 2011). These quality attributes are used to characterise the recurring design concerns, positive consequences, and trade-offs associated with each pattern. Their definitions are provided in Appendix A to ensure a consistent interpretation throughout the catalogue. Additionally, to address RQ2, we also introduce three decision models for selecting the identified design patterns. These decision models guide pattern selection according to the functional requirements of the DT and the specific needs associated with each DT property defined in Section 2. For example, the selection may depend on whether the DT requires direct or indirect observation, continuous, event-driven, or periodic communication, historical data retention for learning or trend analysis, semantic knowledge representation, or bidirectional interaction with the physical twin to achieve a required degree of entanglement. The quality characteristics and subcharacteristics derived from ISO/IEC 25010 represent the potential quality consequences of applying the selected patterns, describing the quality benefits and drawbacks that the resulting DT is expected to exhibit.

6.1 Design patterns for reflection DT property

The *reflection* DT property refers to the ability to assess whether the values of the relevant attributes previously identified through the *representativeness and contextualization* DT property require any transformation before being reflected into the DT state. As discussed in Section 2.2, reflection may involve either a direct mapping of raw measurements or transformations that produce derived information, including analytical, statistical, event-based, and fusion attributes. The three patterns presented in this section collectively cover these alternatives. The Direct Reflection Design Pattern (Section 6.1.1) supports direct mappings when the sensor value can be represented without modification. The Pipeline Design Pattern (Section 6.1.2) supports transformations applied to data obtained from a single source, including analytical computations, statistical aggregations, filtering, normalisation, and the derivation of discrete events. Finally, the Fusion Design Pattern (Section 6.1.3) addresses the integration of multiple homogeneous or heterogeneous sources to derive a single consensus value. The most appropriate pattern must be selected independently for each attribute according to the number of available data sources and the transformation required. Section 6.1.4 shows the decision model for selecting these patterns.

6.1.1 Direct reflection design pattern

Summary An architecture in which raw measurements from the physical twin are directly stored in the DT without any intermediate transformation, ensuring a one-to-one correspondence between physical sensor readings and their digital counterparts.

Context In some DT scenarios, the available sensor data are already expressed in suitable units and formats for the intended analyses and services. In such cases, introducing additional processing stages would add unnecessary complexity and latency without improving the quality or utility of the reflected information. **Running example** (Section 5): this is the case, for example, of environmental sensors monitoring lighting, temperature, or noise levels, where raw readings are directly meaningful for decision-making and visualisation.

Problem When the goal is to keep the DT state as close as possible to the raw measurements of the physical twin, any transformation can distort the original values thus hindering traceability. The challenge is to design a reflection mechanism that preserves the fidelity and immediacy of the data while integrating seamlessly with the DT's event and synchronization mechanisms.

Forces

- Functional correctness: Need for maximum fidelity and traceability between physical measurements and DT state.

Solution The *Direct Reflection design pattern* establishes a direct data path in which *communication agents* forward incoming measurements to *data agents*, which then send them for storage without applying any transformation. Each new sensor reading is immediately written to the corresponding DT attribute, preserving the original value, unit, and timestamp. **Running example** (Section 5): for the environmental attributes (lighting, temperature, noise) we used this reflection pattern because their values do not need any transformation for their representation. Using JaCaMo, *communication agents* are used to retrieve raw values from CArtAgO artifacts and send them to the *data agents*, which store them appropriately in accordance with the memorization patterns (Section 2.3).

Consequences *Benefits*

- Functional correctness: It maximises fidelity between physical and digital values by avoiding unnecessary transformations.*Drawbacks*

- High resource utilization: It may lead to increased storage and bandwidth usage if no additional filtering or compression is applied.
- (Likely) Low functional appropriateness: Raw values may be less directly usable for higher-level analytics, requiring downstream components to perform on-demand transformations.

Related design patterns This pattern can be combined with the *Pipeline Design Pattern* (Section 6.1.2) and *Fusion Design Pattern* (Section 6.1.3) when other DT attributes require additional data transformations or fusions.

Known uses

- Farming: Smart agriculture DTs where soil humidity sensor readings are directly stored in the DT state (Skobelev et al., 2021).

6.1.2 Pipeline design pattern

Summary A agent that collects raw measurements from a physical twin sensor and applies transformation functions, such as statistical aggregations, to produce derived attributes.

Context DTs receive high-frequency sensor data, and directly storing all of these raw data is often infeasible due to bandwidth, latency, and storage constraints. Instead, intermediate agents can perform local aggregation to produce meaningful summaries that enable higher-level reasoning. **Running example** (Section 5): the heart rate readings are aggregated to compute averages, and movement data are mapped to activity levels such as low, medium, or high.

Problem Physical twins generate continuous streams of measurements that require statistical summarization for monitoring dashboards, anomaly detection or trend analysis. In addition, raw data volumes overwhelm storage and real-time processing.

Forces

- Functional appropriateness: Need to modify data without losing essential information.

Solution The problem is addressed by applying the *Pipeline design pattern* (Vermeulen et al., 1995) to process sensor data streams. At regular intervals or once a specific number of measurements have been ingested, these are passed through a sequence of transformation stages (e.g., filtering, normalisation, model-based computation) that culminate in the calculation of derived attributes suitable for storage in the DT. This approach reduces storage requirements and facilitates real-time monitoring and reasoning, while preserving essential behavioural information. **Running example** (Section 5): for the heart rate and movement attributes this pattern was used. Using JaCaMo, *data agents* were developed to obtain data from *communication agents* and apply the required transformations. For the heart rate attribute, the average is calculated for every five readings. Meanwhile, for the movement attribute, the average is calculated for every five numerical values, and this is classified as low, medium or high. These calculated values are stored in the DT in accordance with the memorization patterns (Section 2.3).

Consequences *Benefits*

- Efficient resource utilization: Reduces data transport and storage overhead through localised summarization.
- Functional appropriateness: Facilitates the generation of derived attributes useful for trend or performance analysis.*Drawbacks*

- Modifiability (complexity): Sensor data must be transformed introducing additional effort that increases the implementation complexity, while the transformation pipeline must be maintained as data sources change or transformation rules evolve.

Related design patterns This pattern can be combined with the *Direct Reflection Design Pattern* (Section 6.1.1) and *Fusion Design Pattern* (Section 6.1.3) and when other DT attributes require a direct reflection or a data fusion.

Known uses

- Healthcare: Continuous patient vital sign measurements are aggregated through pipelines that compute time-window averages and statistical summaries for clinical trend monitoring and alerting (Croatti et al., 2020).

6.1.3 Fusion design pattern

Summary A set of agents that cooperate to estimate a single value reflected from multiple data sources, potentially heterogeneous, resolving conflicts and applying priority schemes based on reliability and freshness.

Context Many DT applications employ redundant or heterogeneous sensors monitoring the same physical attribute. Combining these sources improves accuracy and robustness, but introduces challenges related to data synchronisation, unit discrepancies, and conflicting measurements. **Running example** (Section 5): this pattern is not applied in this case study because just one sensor for attribute was used (future extension).

Problem When multiple sensors report values for the same DT attribute, the DT must aggregate them into a single consistent reflected value, handling discrepancies, incomplete data, and varying source reliability without introducing bias or delay.

Forces

- Functional appropriateness: Need to reflect a single meaningful value in the DT, derived from various data sources, to support subsequent analysis and decision-making.

Solution This problem can be solved by implementing an architecture similar to the Blackboard Pattern (Corkhill, 1991). The *Knowledge Sources* (sensors) write their raw timestamped measurements to the shared blackboard as independent contributions. Then, the

Control Component computes the final consensus value using techniques such as weighted voting, Kalman consensus fusion, or majority rule. The resulting fusion value is then forwarded for reflection in the DT state, ensuring a unified representation despite source heterogeneity. **Running example** (Section 5): in the case study, room temperature would be obtained from both a fixed environmental sensor and the wearable device of the child. If, at the same time, the wearable reports 25.1°C and the room sensor reports 23.8°C, the DT cannot directly reflect both values as the current temperature of the room. In this case, the *control component* will evaluate the reliability of each source and compute a consensus value, for example by assigning a higher weight to the fixed room sensor, since it is more representative of the ambient conditions of the room. In this way, conflicting measurements are handled explicitly and a single reflected value is obtained for the DT state.

The *Control Component* may also apply additional transformations over the fused value, for instance to cope with heterogeneous data sources by harmonising units, scales, or formats before the value is stored in the DT (Macías et al., 2024). Such processing is particularly relevant when combining readings from sensors with different levels of quality, accuracy, or reliability, as appropriate fusion and normalisation are essential to obtain meaningful and trustworthy information from both homogeneous and heterogeneous sources. Nevertheless, these transformations are not mandatory, as in some scenarios it is preferable to store the fused value in its raw form.

Consequences *Benefits*

- Functional correctness: Provides a single, accurately reflected attribute, constructed from multiple views of the same physical phenomenon.
- Fault tolerance: Increases resistance to temporary failures and noise from individual sensors.*Drawbacks*

- High resource utilization: Increases computational and implementation complexity, especially when the number of sources grows.

Related design patterns This pattern can be combined with the *Direct Reflection Design Pattern* (Section 6.1.1) and *Pipeline Design Pattern* (Section 6.1.2) when other DT attributes require a direct reflection or some data transformations.

Known uses

- Smart Traffic: Multi-camera traffic flow DTs where position estimates from overlapping views are aggregated using confidence-weighted voting to track vehicles more accurately than individual cameras (Li & Zhang, 2023)

6.1.4 Decision model for reflection DT property

The decision model of Fig. 5 is designed for selecting the reflection design patterns presented previously. This decision model must be applied independently for each DT attribute,

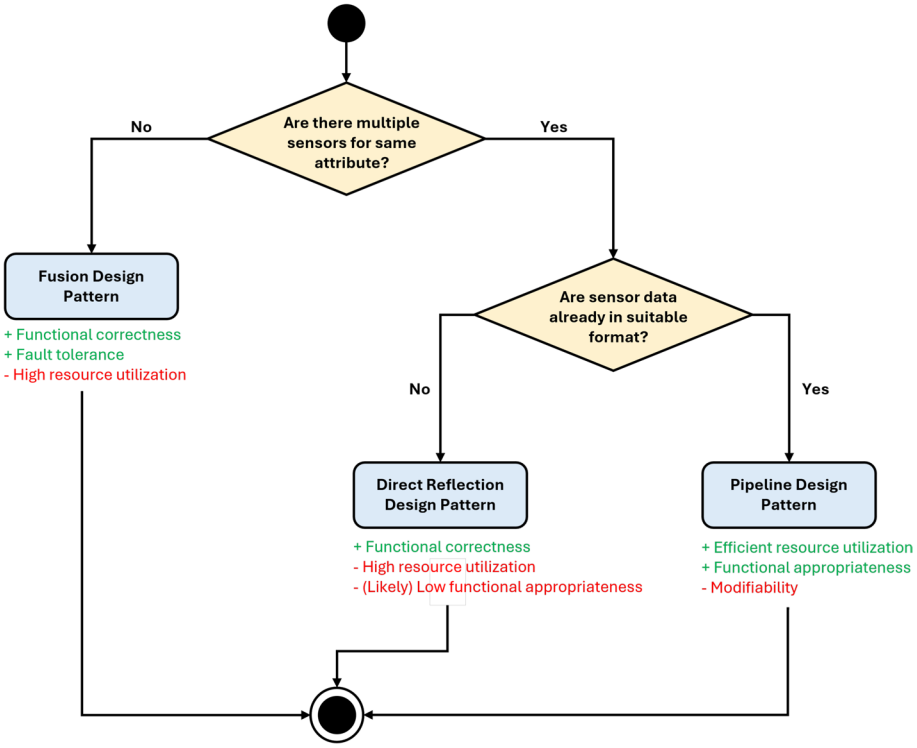


Fig. 5 Decision model for reflection design pattern selection

since each one may require distinct reflection strategies depending on their data sources and processing requirements. In the following, each step of the decision process is detailed.

Step 1: Determine the number of data sources The first step evaluates whether multiple sensors or heterogeneous data sources provide measurements for the same physical attribute.

- **Multiple sources**: When several sensors monitor the same attribute, the DT must reconcile discrepancies and produce a consolidated representation. In such cases, the *Fusion Design Pattern* (Section 6.1.3) should be applied, as it provides a structured mechanism for integrating multiple data sources and deriving a consensus value.
- **Single source**: If only one data source is available for the attribute, proceed to Step 2.**Running example** (Section 5): As each attribute (heart rate, movement, lighting, temperature, noise) uses a single sensor, we proceed to the next step.

Step 2: Transformation evaluation The second step determines whether the data obtained from the single source requires a transformation before it can be reflected in the DT state.

- **Transformation needed**: When the processing is necessary, the *Pipeline Design Pattern* (Section 6.1.2) is recommended, as it enables the sequential execution of processing stages that transform raw sensor data into the desired attribute representation.

- **No transformation**: When the data can be directly mapped to the DT state without additional processing, the *Direct Reflection Design Pattern* (Section 6.1.1) provides a simpler and more efficient mechanism for updating the corresponding DT attribute.**Running example** (Section 5): As the values of the heart rate and movement require transformations, we applied the *Pipeline Design Pattern* (Section 6.1.2) for them. Moreover, for the values of the environmental attributes we selected the *Direct Reflection Design Pattern* (Section 6.1.1) because no transformations are required.

6.2 Design patterns for memorization DT property

Once the relevant attributes have been identified through *representativeness and contextualization* DT property, and their values have been expressed in appropriate terms through *reflection* DT property, the next step is to decide how the resulting information should be stored in the DT. This is the role of the *memorization* DT property. As described in Section 2.3, memorization encompasses both dynamic and static information. Dynamic information information refers to attributes whose values change over time. Static information may concern structural, configurational, or semantic aspects of the physical twin. The four patterns presented in this section collectively address all these forms of memorization. The Current-State Design Pattern (Section 6.2.1) supports the storage of the latest value of dynamic attributes for real-time reasoning, whereas the Time-Series Data Design Pattern (Section 6.2.2) supports the persistence of their historical evolution. The Frame Model Design Pattern (Section 6.2.3) represents static structural and configurational information, while the Ontology Model Design Pattern (Section 6.2.4) aims at memorizing semantic concepts and relationships. Section 6.2.5 shows the decision models for selecting them.

6.2.1 Current-state design pattern

Summary This pattern is used to represent the *dynamic attributes* (Section 2.3) of the physical twin when only their latest value is required for real-time decision-making. According to this pattern, the value of such attributes is stored as current facts or beliefs in the agent's knowledge base enabling its direct use by rule-based or goal-oriented decision mechanisms.

Context This pattern applies to DTs whose objective is real-time monitoring, alarm triggering, or immediate fault detection, without the need to retain historical data of the underlying attributes. Typical examples include threshold-based alarms, interlocks, or safety checks where decisions depend solely on the current operating condition rather than on long-term trends. **Running example** (Section 5): this pattern is used for dynamic attributes (heart rate, movement, lighting, temperature, noise), whose latest values enable real-time decisions such as adapt the duration of the study session.

Problem Persisting full historical time-series for every dynamic attribute increases storage and processing requirements, even when the behaviour of the DT only depends on the most recent value. In such cases, producing historical data for them adds complexity without improving decision quality. A more compact representation is needed, focused on the current state that the inference mechanism must reason about.

Forces

- Time behaviour: The DT must react quickly to present condition. For example, when a safety threshold is crossed, the current value must be directly accessible in the knowledge base.
- Resource utilization: Historical information is not required by the specified use cases. Since no trend analysis, ML training or long-term prognosis needs to be conducted, storing this information would be an unnecessary overhead.

Solution The current value of each dynamic attribute should be represented in the knowledge base of agents that composed the DT. As new sensor readings arrive, the corresponding beliefs are updated in place, ensuring that the knowledge base always reflects the latest state. Decision rules then operate on these current-state beliefs to trigger alarms, select actions, or update intentions, without consulting any historical record. **Running example** (Section 5): the current value of each dynamic attribute (heart rate, movement, lighting, temperature, noise) is represented as beliefs in the knowledge base of *decision-making agent* implemented in Jason. Therefore, *data agents* continuously send values to its knowledge base for immediate rule-based reasoning.

Consequences *Benefits*

- Efficient resource utilization: Minimises storage and computational overhead by keeping only the latest values of dynamic attributes instead of a full time-series.
- Functional appropriateness: Simplifies rule-based reasoning, since decision rules operate directly over a compact and up-to-date set of facts or beliefs.*Drawbacks*

- (Likely) Low functional completeness: Looses access to historical data, preventing the use of trend-based analysis or ML models that require sequences of past observations.

Related patterns It can be combined with the *Time-Series Data Design Pattern* (Section 6.2.2) if the historial storage is needed for other objectives of the DT. Moreover, it can also be combined with the *Frame Model Design Pattern* (Section 6.2.3) and *Ontology Model Design Pattern* (Section 6.2.4), if the DT needs to store static information about the physical twin.

Known uses

- Energy: DT for UAV-assisted mobile edge computing in energy-constrained mobile networks, where real-time state variables are stored as current facts in the agent knowledge base to support immediate decision-making (Liu et al., 2023).

6.2.2 Time-series data design pattern

Summary This pattern is used to represent the dynamic attributes of the physical twin whose historical values are required, for instance to train and apply machine learning (ML) models or for visualisation. In this pattern, the knowledge related to such attributes

is stored as time-series data, ensuring that the DT can effectively capture and exploit the temporal evolution and trends.

Context This pattern applies to DTs in which certain dynamic attributes must be retained over time because their historical evolution is used, for example, for later data visualisation or to build and update ML models for tasks such as prognosis, anomaly detection, and predictive maintenance. **Running example** (Section 5): it is necessary to store the data history for post-session visualisation by tutors.

Problem Representing dynamic attributes as isolated frames removes the temporal structure needed for visualisation and learning-based analyses. If the DT only keeps current states, it cannot capture changes or trends over time.

Forces

- Functional appropriateness: Some DT tasks, such as predictive maintenance or behaviour predicting, require ML models trained on sequences of past observations, not on single-point summaries.

Solution The dynamic attributes that require historical data should be represented as time-series where each attribute is stored as a sequence of timestamped values in a dedicated *Time Series Database* (Rudakov et al., 2023) acting as an external repository for historical numeric data. Thus, other components of the DT can query this external store directly to obtain training data or inference inputs over specific time windows. **Running example** (Section 5): as JaCaMo does not inherently support the storage of historical data, an external repository is used to store these historical data. For this aim, the *data agents* send values with timestamps to a MongoDB database (MongoDB, 2026). Then, the *reporting agent* can access to these data to generate the reports for the tutor.

Consequences *Benefits*

- Functional completeness: Preserves the full temporal resolution required to for example train effective ML models, enabling detection of trends, patterns, and anomalies.*Drawbacks*
- High resource utilization: DT may require additional storage and computing resources when managing high-frequency streams over long periods.

Related design patterns It can be combined with the *Current-State Fact Design Pattern* (Section 6.2.1) when the DT also needs to support real-time decision-making. Moreover, it can also be combined with the *Frame Model Design Pattern* (Section 6.2.3) and *Ontology Model Design Pattern* (Section 6.2.4), if the DT needs to store static information about the physical twin.

Known uses

- Industry: A DT for simulating organisational processes, in which time-series data are stored externally for simulation and training (Lektauers et al., 2022).

- Health: Physiological time-series data are stored in the DT to later train ML algorithms for risk prediction (Croatti et al., 2020).
- Energy: DT for UAV-assisted mobile edge computing in energy-constrained mobile networks, where historical traces are collected to be used by deep reinforcement learning models (Liu et al., 2023).

6.2.3 Frame model design pattern

Summary The *Frame Model Design Pattern* is a design pattern for representing structural and configurational static attributes of the physical twin within the agents that compose the DT implemented on a MAS architecture. This pattern establishes how the relevant structure and configuration of the physical twin are formalised as frames in the knowledge bases of agents.

Context Sometimes certain relevant attributes of the physical twin remain static. These attributes do not change over the time, but are essential for the proper functioning of the DT. In such cases, agents that composed the DT need to access this static information to support their reasoning, monitoring, and control capabilities. **Running example** (Section 5): the age of the child (structural attribute) and the constraint that no more than 2 messages can be sent per minute (configurational attribute) are relevant static data that the DT must take into account when making decisions.

Problem DTs require a stable representation of the physical twin's static attributes. In MAS-based DTs, agents must be capable of reasoning about the value of these attributes to interpret sensor data, validate physical configurations, and enforce constraints. The challenge lies in defining a model that captures all relevant static attributes in a way that integrates seamlessly with agent reasoning mechanisms while remaining extensible and domain-independent.

Forces

- Time behaviour: Each agent requires fast access to static structural and configurational attributes.
- Modularity: The representation should be modular and scalable, enabling reconfiguration as the physical twin evolves or changes.

Solution Represent the static information of the physical twin as *frames* (Mehdi, 2025) encoded within each knowledge base of agents. Each frame contains *slots* that describe attributes along with their values and constraints. This structure enables each agent to maintain a coherent and semantically organised depiction of the static attributes of the physical twin, facilitating reasoning, validation, and synchronisation actions within the MAS-based DT architecture. **Running example** (Section 5): The age of the child and the limit of two messages per minute are implemented as frames within the knowledge base of the *decision-making agent*, providing a stable context for its reasoning based on personalised rules.

Consequences *Benefits*

- Modularity: Provides a formal and modular structure for representing static attributes.
- Functional appropriateness: Ensures structural consistency between the digital and physical entities within the DT.*Drawbacks*
- (Likely) Low functional completeness: Offers limited reasoning support, as frames capture structural attributes but do not enable automated inference over shared domain knowledge.

Related patterns This pattern can be complemented by the *Ontology Model Design Pattern* (Section 6.2.4) to provide semantic meaning to the structural and configurational information. Additionally, it can also be combined with the *Current-State Design Pattern* (Section 6.2.1) and *Time-Series Design Pattern* (Section 6.2.2), if the DT needs to store dynamic information about the physical twin.

Known uses

- Energy: In a DT of an electric substation, the structural information is organised into frames that provide a stable representation of the physical infrastructure, supporting the coordination between agents and the intelligent functionality of the DT (Yuan et al., 2022).

6.2.4 Ontology model design pattern

Summary The *Ontology Model Design Pattern* is a design pattern for representing the static knowledge of the physical twin in a MAS-based DT when this knowledge includes *semantic attributes* such as domain knowledge or relationships between components that compose the physical twin. In this pattern, this semantic knowledge is captured in an ontology that can agents query and instantiate, enabling semantic interoperability and automated reasoning over the conceptual and relational structure of the physical twin.

Context This pattern applies to DTs where the static knowledge of the physical twin encompasses not only structural and configurational attributes but also semantic relationships or domain-specific concepts that agents that compose the DT must interpret consistently. Agents require access to a shared conceptual model to coordinate actions, validate configurations against relational constraints, and reason about the implications of changes across interconnected components of the physical twin. **Running example** (Section 5): this pattern does not need to be applied in this case study because there is no relevant semantic information to represent in the DT.

Problem When static knowledge includes some semantic attributes, representing them solely through local data structures or frames leads to semantic discrepancies between agents. Each agent may interpret concepts and relationships differently, which hinders global reasoning and the detection of cascading effects or configuration incompatibilities. A formal, shared representation of this semantic knowledge is essential to ensure interoperability and enable inference about the conceptual structure of the physical twin.

Forces

- Functional completeness): The DT must reason over semantic attributes of the physical twin and its domain.
- Interoperability: The representation must be modifiable to adapt to the evolution of the domain and the relationships between the components of the physical twin.

Solution Define a *ontology* (Pan et al., 2003) to formally capture the semantic attributes of the physical twin. The ontology comprises *classes* representing components of the physical twin and domain concepts and *properties* for describing relationships between them. Agents query the ontology to instantiate the static knowledge locally to perform semantic validation, infer relational implications, and coordinate actions based on the shared conceptual model of the physical twin. **Running example** (Section 5): the case study would incorporate a shared ontology to represent concepts such as the *child*, *tutor*, *study session*, *sensor*, *actuator*, and *intervention*. The semantic relationships among these concepts will be complemented by a Moise organisational specification within JaCaMo (Fig. 4), which will manage the roles, goals, and norms governing agent interactions. This will enable agents to interpret domain concepts consistently while coordinating their responsibilities and interactions according to the organisational constraints of the study session management scenario.

Consequences *Benefits*

- Interoperability: Ensures semantic consistency across agents by providing a shared, formal vocabulary and a common understanding of static aspects.
- Functional completeness: Supports automated reasoning and consistency checking over relationships and configurations, reducing the need for ad-hoc rule duplication.*Drawbacks*

- Modifiability: Designing, maintaining, and evolving the ontology may require substantial effort when domain concepts, relationships, or constraints change.
- Low performance efficiency (resource utilization and time behaviour): Reasoning over large semantic models may increase processing time and computational resource consumption, particularly in real-time applications.

Related Patterns This pattern complements the *Frame Model Design Pattern* (Section 6.2.3) by providing semantic relationships where frames handle purely structural attributes. Also, it can also be combined with the *Current-State Design Pattern* (Section 6.2.1) and *Time-Series Design Pattern* (Section 6.2.2) when the DT needs to store dynamic information about the physical twin.

Known uses

- Farming: DT uses ontologies to model specific knowledge about plant growth and environmental factors that influence growth dynamics, with the aim of predicting the conditions under which plants will grow best (Skobelev et al., 2021).

- Health: DT uses ontologies to support resource allocation by determining which hospital resources should be mobilised for a given clinical situation (Mammadova & Ahmadova, 2023)

6.2.5 Decision models for memorization DT property

The selection of the memorization design patterns depends on the nature of the attributes: static or dynamic. Therefore, two complementary decision models are designed: one for static attributes (Fig. 7) and another for dynamic attributes (Fig. 6). These decision models can be applied independently or in combination to identify the necessary knowledge representations according to the requirements of each type of attribute represented in the DT.

Step 1: Decision model for dynamic attributes Dynamic attributes represent time-varying observable states captured through sensors. The decision model showed in Fig. 6 guides developers in selecting the most appropriate representation of dynamic attributes, depending on whether they need to retain only the current value or keep the history.

- **Current Data**: Select the *Current-State Fact Design Pattern* (Section 6.2.1). The current value of the dynamic attributes are stored directly in the knowledge base of agents, enabling direct rule-based reasoning. This pattern minimises storage overhead by retaining only the latest sensor readings.
- **Historical Data**: Select the *Time-Series Data Design Pattern* (Section 6.2.2). Dynamic attributes are stored as timestamped sequences in external time-series databases. This pattern supports querying over specific time windows for machine learning model training, trend analysis, visualisation, and prognostic.
- **Current and Historical Data**: When both immediate decision-making and predictive capabilities are required, select both patterns: *Current-State Fact Design Pattern* (Section 6.2.1) and *Time-Series Data Design Pattern* (Section 6.2.2).**Running example** (Section 5): The values of all the attributes (heart rate, movement, lighting, temperature, and noise) are required for both real-time decision-making and post-session visualisation, leading to the selection of the both patterns: *Current-State Fact Design Pattern* (Section 6.2.1) and *Time-Series Data Design Pattern* (Section 6.2.2).

Step 2: Decision model for static attributes The decision model, showed in Fig. 7, determines if static knowledge (structural, configurational, and semantic attributes of the physical twin) should be represented within each knowledge base of agents or through a shared ontology across the MAS.

1. **Structural and/or configurational attributes**: Select the *Frame Model Design Pattern* (Section 6.2.3). This pattern organises static information as frames containing attribute values and constraints, stored in the knowledge base of agents. Frames are ideal for agents that require fast access to static knowledge without global semantic alignment.
2. **Semantic attributes (domain knowledge, relationships)**: Select the *Ontology Model Design Pattern* (Section 6.2.4). This pattern formalises static knowledge as a shared ontology with classes representing entities and properties capturing relationships.

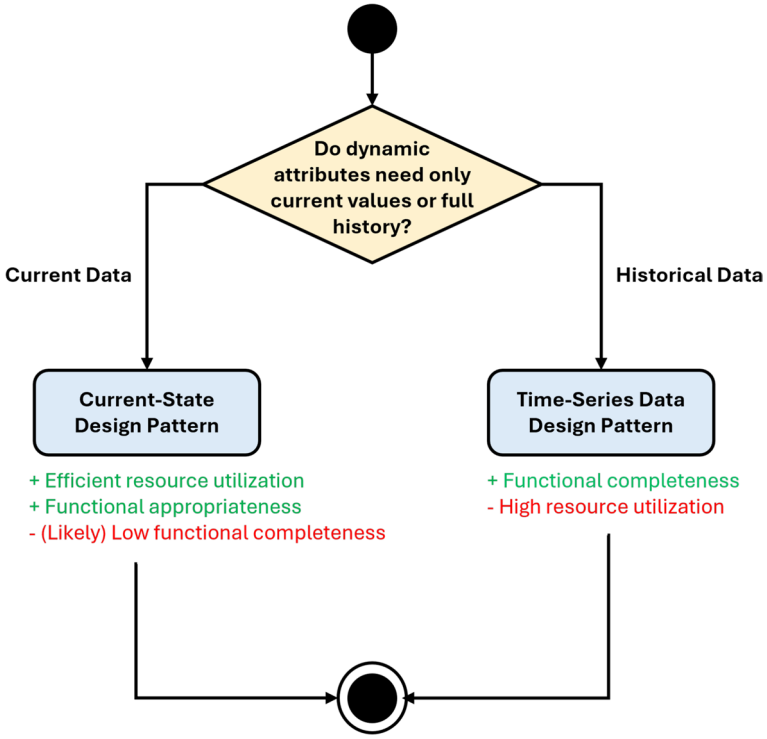


Fig. 6 Decision model for representation of dynamic attributes (memorization)

Ontologies ensure semantic interoperability across agents, enabling automated reasoning over dependency chains, configuration validation, and conceptual consistency.**Running example** (Section 5): Since no semantic knowledge needs to be represented, just the *Frame Model Design Pattern* (Section 6.2.3) is used to represent the age of the child and the constraint of not sending more than two messages per minute.

6.3 Design patterns for entanglement DT property

The *entanglement* DT property concerns the degree and direction of coupling between the physical twin and its DT. As explained in Section 2.4, entanglement ranges from a weak entanglement, in which no direct interaction is established, to simple entanglement, in which information flows in one direction, and strong entanglement, in which information is exchanged bidirectionally and the DT may control the behaviour of the physical twin. In addition, the communication from the physical twin to the DT may require continuous, event-driven, or periodic updates depending on the required promptness. The patterns presented in this section collectively cover these alternatives. The Indirect Observation Design Pattern (Section 6.3.1) supports weak entanglement through the observation of the physical twin by an intermediary. The Continuous Direct Data Dispatcher (Section 6.3.2), Event-Driven Direct Data Update (Section 6.3.3), and Periodic Direct Data Sending (Section

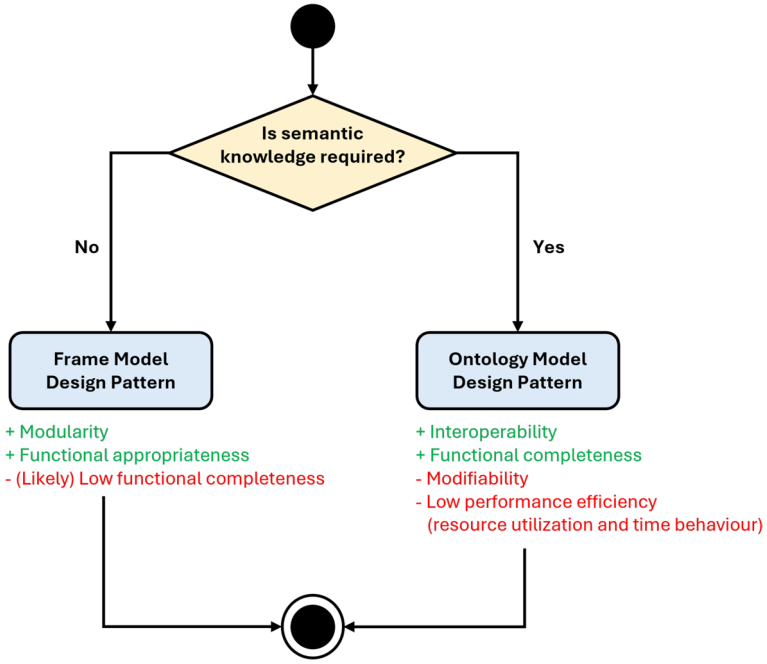


Fig. 7 Decision model for representation of static attributes (memorization)

6.3.4) patterns support direct physical to digital communication under different promptness requirements. Finally, the Action Dispatcher Design Pattern (Section 6.3.5) supports digital to physical communication enabling the bidirectional interaction between both. Section 6.3.6 shows the decision model for selecting them.

6.3.1 Indirect observation design pattern for weak entanglement

Summary This pattern implements an agent that infers the state of a physical twin indirectly by analysing data from its surrounding environment or external tools and periodically sends information to the agents of the DT to update its state. This pattern is characteristic of *weak entanglement*.

Context In some cases, physical twins lack integrated sensors or cannot establish a direct communication channel with their DT. However, their state can be inferred by observing their surrounding environment. For example, the state of a physical twin can be monitored by a camera, extracting relevant information and transmitting it to the DT. **Running example** (Section 5): this pattern does not need to be applied because there are sensors that provide the information directly for all relevant attributes.

Problem How to synchronize the DT when there is no direct connection with its physical twin?

Forces

- Functional correctness: The inferred state must correctly represent the physical twin based on the available environmental or external data.

Solution This problem can be solved by implementing an architecture similar to the MAS Observer Pattern (Mohamad et al., 2006). First, it is necessary implementing a *subject agent* that observes the environment or analyses some external data sources related to the physical twin. This agent processes these data and transmits the information to the *observer agents*, which are in the DT. As the subject agent has to infer the state of the physical twin, it will transmit data periodically rather than in real time or when some event occurs. **Running example** (Section 5): the case study would incorporate a camera-based observation agent to determine whether the child is present in the study area. This agent will process the visual information and infer a presence state, such as *seated* or *not seated*.

When multiple heterogeneous external sources are available, such as cameras, social networks, public APIs, etc., the subject agent is implemented following the MAS Aggregation Pattern (Fernandez-Marquez et al., 2011) to collect, integrate and harmonise the data before sending it to the DT. This improves the accuracy of the inferred state of the physical twin.

Consequences *Benefits*

- Functional appropriateness: Agents need to infer the physical twin status from environmental data.
- Resource utilization: Enables the DT to monitor physical twin without intrusive or costly sensors.*Drawbacks*

- (Likely) Low functional correctness: The DT state may provide less correct results, or results with lower precision, than a state based on direct measurements, because it is inferred from indirect observations.

Known uses

- Smart city: DT for monitoring population movement using cameras (Yun et al., 2020).

6.3.2 Continuous direct data dispatcher design pattern for simple and strong entanglement

Summary This pattern implements an internal agent into the physical twin that continuously and directly monitors and transmits the state of the physical twin and any relevant data to the DT through appropriate networks and communication protocols. This pattern is characteristic of *strong entanglement* supporting the direct continuous link from the physical twin to the DT. In addition, this pattern is also characteristic of *simple entanglement* of DTs whenever continuous communication from the physical twin to the DT is required.

Context Sometimes, the physical twin has embedded computational and networking capabilities to communicate continuously and directly its information to its DT. This occurs in smart devices with sensors that report their status without relying on external observation. **Running example** (Section 5): it is necessary to continuously monitor the heart rate and movement of the child in order to assess his state in real time.

Problem How to synchronize the DT when the physical twin has direct and continuous communication capabilities?

Forces

- Interoperability: The physical twin and the DT must be able to exchange information directly through compatible sensors, networks, communication protocols, and message formats.
- Time behaviour: The DT requires continuous state updates within the response and processing times required for real-time monitoring.

Solution An agent is implemented in the physical twin that collects data and transmits it directly to the DT. This can be implemented by using the MAS Publish-Subscribe Pattern (Chella et al., 2010). Here, the *publisher agent* is responsible for collecting and sending data in real time to the *message broker*. This broker is responsible for retransmitting messages to the *subscriber agents*. Each agent is subscribed to relevant topics, thus receiving only useful information. This architecture with an intermediary allows for a reduction in message traffic, which is essential for efficient continuous communication. **Running example** (Section 5): as the JaCaMo framework is used for the implementation, some CArtAgO artifacts act as the message broker. Thus, these artifacts receive real-time sensor data from the physical twin (publisher). Then, the *communication agents* (subscriber) collect and send them to *data agents* for proper processing and storage.

Consequences *Benefits*

- Functional correctness: Supports a DT state that correctly represents the physical twin by transmitting direct measurements without relying on indirect inference.
- High availability: it supports continuous access to current physical-twin data in the DT under normal operating conditions, while the direct communication channel and embedded agent remain operational.*Drawbacks*
- High resource utilization: The timely availability of current DT data depends on the continuous operation of the communication network and its supporting infrastructure.

Related design patterns It should be combined with the Action Dispatcher Pattern to enable a DT with *strong entanglement*.

Known uses

- Manufacturing: industrial machinery with embedded sensors that update their DT (Gayathri & Uma, 2018).

- Healthcare: medical devices that send data to the DT of the patient (Heininger & Stary, 2021).

6.3.3 Event-driven direct data design pattern pattern for simple entanglement

Summary This pattern implements an agent in the physical twin that responds to specific events to directly send information to agents in the DT to update its state. This pattern is characteristic of *simple entanglement* of DTs that require event-driven communication from the physical twin to the DT.

Context In some cases, it is not necessary to continuously send data from the physical twin, but it is sufficient to update the state of the DT whenever a relevant event occurs in the physical twin. **Running example** (Section 5): this pattern was not applied because the child attributes (heart rate, movement) require continuous real-time monitoring and the environmental attributes (lighting, temperature, noise) are collected periodically rather than event driven.

Problem How to ensure that the DT is updated whenever relevant events occur in the physical twin?

Forces

- Resource utilization: The physical-twin agent must minimise network, processing, and energy resource consumption by transmitting data only when relevant events occur.
- Functional correctness: The physical-twin agent must correctly detect the predefined events that require the DT to be updated.

Solution The architecture of the MAS Event Dispatcher pattern (Deugo et al., 1999) is used to provide a solution to this problem. This pattern defines explicit agreements between agents on which events should be notified and how they should be handled. So, an *event-driven reactive agent* is implemented in the physical twin. This agent continuously monitors the state of the physical twin and detects predefined events. When an event occurs, the event-driven reactive agent uses the architecture of MAS Observer pattern (Mohamad et al., 2006) to notify the DT by sending the new information to the *observer agents* of the DT. Thus, the event-driven reactive agent acts as a *subject agent*. **Running example** (Section 5): this pattern would be used to avoid continuous heart rate monitoring by transmitting data to the DT only when relevant events occur, such as a heart rate value exceeding an individualised threshold. This approach will reduce unnecessary communication and processing while ensuring that potentially relevant changes in the child's physiological state are promptly reported to the DT.

The use of the MAS Event Dispatcher pattern (Deugo et al., 1999) allows the event-driven reactive agent to dynamically decide whether to send the information to the DT or not, depending on the criticality of the event and the availability of resources.

Consequences *Benefits*

- Functional appropriateness: The DT is immediately updated in case of critical events.
- Resource utilization: The agent into the physical twin minimises the use of resources by sending data only when necessary.*Drawbacks*

- (Likely) Low functional correctness: If a relevant event is not detected, the DT does not receive the information required to update its state for that event.

Known uses

- Farming: data collected from plants are sent to the DT once a day or whenever certain events occur (Skobelev et al., 2021).

6.3.4 Periodic direct data sending design pattern for simple entanglement

Summary This pattern implements an agent in the physical twin that sends periodic status updates to the DT. It uses a model based on fixed time intervals to ensure that the DT receives information on a regular basis. This pattern is characteristic of simple entanglement of DTs that require periodic communication from the physical twin to the DT.

Context In some cases, the physical twin does not experience abrupt or critical changes, but it is important to keep constant synchronisation with the DT. Therefore, a periodic communication mechanism is necessary. **Running example** (Section 5): data of the environmental attributes (lighting, temperature, noise) are collected periodically every 1 minute.

Problem How to ensure that the DT receives regular updates from the physical twin without overloading the network?

Forces

- Time behaviour: The physical-twin agent must send state updates at predefined intervals that satisfy the required update frequency of the DT.

Solution The architecture of the MAS Observer Pattern (Mohamad et al., 2006) is used to provide a solution to this problem. First, it is necessary implementing a *subject agent* that periodically collects the data from the physical twin and sends this information to the *observer agents* at predefined time intervals. **Running example** (Section 5): the environmental sensors send data to our CArtAgO artifacts every minute (subjects). Then, the *communication agents* of the DT retrieve these values (observers) and send them to the *data agent* for proper processing and storage.

Consequences *Benefits*

- Functional appropriateness: Provides regular DT state updates at the specified periodic intervals.
- Resource utilization: Can reduce network and processing resource consumption compared with continuous data transmission.*Drawbacks*

- (Likely) Low functional correctness: Between periodic transmissions, the DT state may no longer correctly represent the most recent state of the physical twin.
- (Likely) High resource utilization: May consume unnecessary network, processing, or storage resources when the update interval is shorter than required by the rate of significant physical-state changes.

Known uses

- Health: Medical devices that alert to changes in the health status of the patient (Croatti et al., 2020).

6.3.5 Action dispatcher design pattern for simple and strong entanglement

Summary This pattern allows agents in the DT to send commands to the physical twin, allowing its behaviour or state to be modified by actuators. This pattern is characteristic of strong entanglement supporting the link from the DT to the physical twin. In addition, this pattern is also characteristic of simple entanglement of DTs that require communication from the DT to the physical twin.

Context Occasionally, the DT not only needs to reflect the state of the physical twin, but also to interact with it, emitting decisions that translate into physical actions, such as activating an alarm, adjusting a parameter or turning on a device. **Running example** (Section 5): in this case, the DT communicates with the child, tutor, and environment through targeted messages. For the child, it delivers alerts via the smart band such as reminders to take breaks. For the tutor, it provides detailed session reports. Environmental actuators receive commands to adjust lighting and temperature for optimal study conditions.

Problem How to enable the DT to communicate instructions to its physical twin to produce real changes in the physical world?

Forces

- Functional completeness: The DT can send commands to modify the behaviour of the physical twin

Solution This pattern can be implemented by using several MAS design patterns together. First, an agent implemented in the DT is required following the MAS Command Dispatcher Pattern (Dupire & Fernandez, 2001), which encapsulates the action as a message with some

defined parameters. Then, to send messages from the DT and to its physical twin, the MAS Publish-Subscribe (Chella et al., 2010) can be used. Thus, an agent of the DT (*publisher agent*) sends the generated message to an intermediary (*message broker*). Finally, in the physical twin, for each actuator, an agent (*subscriber agent*) is implemented as a message listener that translates messages into actions for their execution. **Running example** (Section 5): on the one hand, the *decision-making agent* and the *reporting agent* act as publishers. They send personalised messages and reports, respectively, to the CArtAgO artifacts, which operate as message brokers. These artifacts manage and distribute the information to the corresponding subscribers. In this case, the smart wristband subscribes to the artifacts to receive personalised messages for the child, the tutor receives by email the reports, and the environmental actuators (lighting and temperature control) receive commands to dynamically adjust the environmental conditions.

Consequences *Benefits*

- Functional appropriateness: Enables the DT to perform control functions that modify the physical twin behaviour in accordance with decisions made by the agents of the DT.*Drawbacks*
- Modifiability: The DT requires a more advanced architecture.
- (Likely) Low integrity: The use of remote control could introduce security risks or errors.

Related design patterns It should be combined with the Continuous Direct Data Dispatcher to enable a DT with strong entanglement.

Known uses

- Energy: Smart homes that optimise their energy consumption according to conditions detected by the DT (Huang et al., 2023).
- Manufacturing: Robots controlled in real time by a DT (Glaser & Strauss, 1999).

6.3.6 Decision model for entanglement DT property

This decision model is designed for selecting the previous entanglement design patterns according to the three foundational aspects of entanglement: connectivity, promptness and association. These aspects allow to systematically classify the degree of entanglement into weak, simple, or strong, as illustrated in Fig. 8. Notably, multiple communication patterns from the physical twin to the DT can be applied simultaneously, depending on the specific requirements of each relevant attribute.

Step 1: Determine the type of connectivity (C) The first step evaluates the nature of the connection between the physical twin and the DT. Two possible outcomes are considered:

- **Indirect:** The communication between the physical twin and its DT is mediated by intermediaries. In this case, the entanglement is classified as *Weak Entanglement* due to the limited and indirect interaction, and the recommended pattern is *Indirect Observation Design Pattern* (Section 6.3.1).

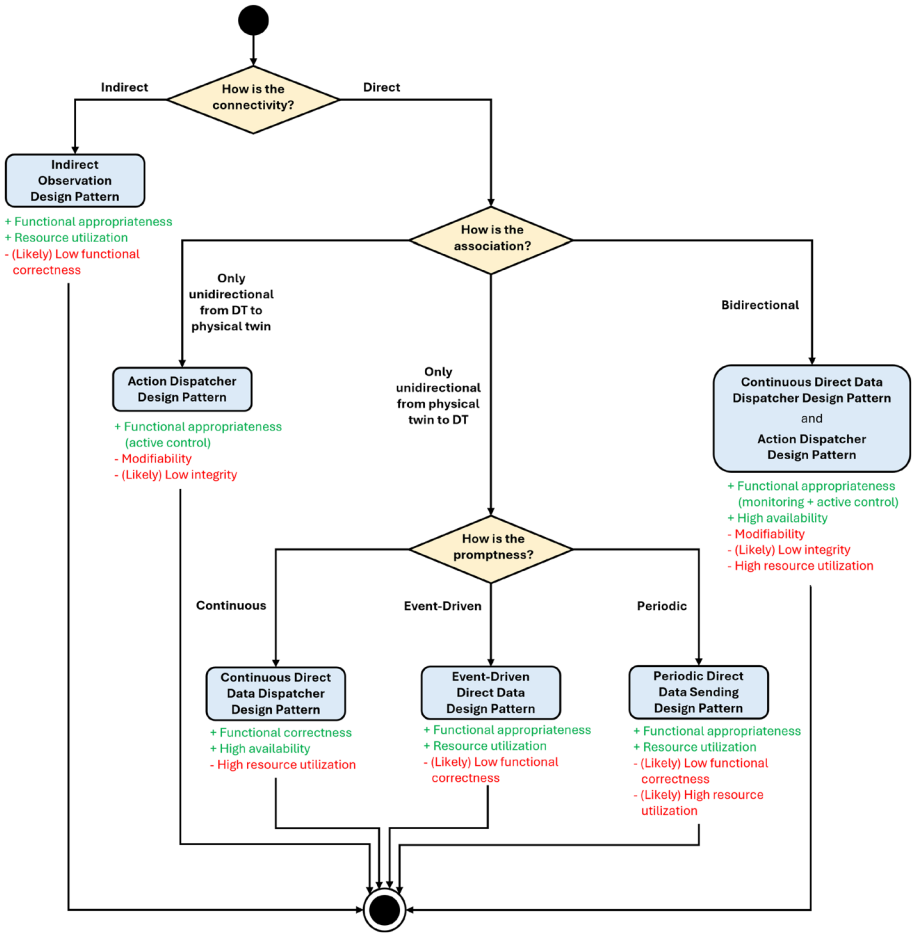


Fig. 8 Decision model for entanglement design pattern selection

- **Direct:** The physical twin and DT can communicate directly without intermediaries. In this case, the decision process continues to the second step to determine the requirements of association for the DT.**Running example** (Section 5): as there are sensors that send information directly about the child and his environment, we proceed to step 2.

Step 2: Determine the direction of association (A) The second step examines the directionality of communication between the physical twin and the DT. Three possible scenarios are considered:

- **Bidirectional:** Information flows in both directions, enabling a dynamic interaction between the physical twin and its DT. In this case, the entanglement is classified as *Strong Entanglement*. The recommended patterns are the *Action Dispatcher Design Pattern* (Section 6.3.5) for the communication from the DT to the physical twin and for the com-

munication from the physical twin to the DT, the decision process should continue to the third step to determine the promptness requirements of each attribute.

- **Unidirectional from DT to physical twin:** Information flows exclusively from the DT to the physical twin. This scenario results in *Simple Entanglement*, as the interaction is limited to one direction. The recommended pattern in this case is the *Action Dispatcher Design Pattern* (Section 6.3.5).
- **Unidirectional from physical twin to DT:** Information flows exclusively from the physical twin to the DT. Similar to the previous scenario, this also results in *Simple Entanglement*. However, in this case, the decision process should continue to the third step to determine the promptness requirements when communicating from the physical twin to the DT.**Running example** (Section 5): in this case study, the DT sends commands to the smart wristband, tutor notifications, and environmental actuators while simultaneously receiving sensor data, establishing a bidirectional association that classifies the entanglement as *strong entanglement*. Therefore, the *Action Dispatcher Design Pattern* (Section 6.3.5) is selected for the communication from the DT to the physical twin. Then, for the communication from the physical twin to the DT, the decision process should continue to the step 3 to determine the promptness requirements of each attribute.

Step 3: Determine the required promptness (P) The third criterion evaluates the timing and frequency of the interactions. Three possible outcomes are considered:

- **Continuous:** Data is sent from the physical twin to the DT instantaneously, ensuring that changes in the physical twin are immediately reflected in the DT. The recommended pattern is *Continuous Direct Data Dispatcher Design Pattern* (Section 6.3.2).
- **Event-Driven:** Data sending occurs in response to specific events or triggers, but not continuously. The recommended pattern is the *Event-Driven Direct Data Update Design Pattern* (Section 6.3.3).
- **Periodic:** Data sending occurs at predefined regular intervals. The recommended pattern is the *Periodic Direct Design Pattern* (Section 6.3.4).**Running example** (Section 5): in this case study, the child attributes (heart rate and movement) require continuous data transmission, so the selected pattern for them is the *Continuous Direct Data Dispatcher Design Pattern* (Section 6.3.2), while environmental attributes (lighting, temperature, and noise) are collected periodically every 1 minute using the *Periodic Direct Data Sending Design Pattern* (Section 6.3.4).

7 Threats to validity

Several threats to the validity of this study were identified and are discussed below (Zhou et al., 2016):

- *Construct Validity*: The definitions of the *reflection*, *memorization* and *entanglement* DT properties are based on the work of Minerva et al. (2020). Although this work is widely accepted (citations: 506; source: Scopus; date: March 2026), it remains subject

to interpretation. On the other hand, a potential threat to validity is researcher subjectivity during the coding and interpretation of the selected studies. To mitigate this risk, the initial coding was performed by the first author, while the resulting codes, categories, and patterns were subsequently reviewed by the remaining authors, and disagreements were resolved through consensus.

- *Internal Validity*: The identification of patterns may have been influenced by the bias of the authors of this paper during the analysis process. To mitigate this threat, all patterns were cross-checked against existing MAS literature.
- *External Validity*: The applicability of the patterns was demonstrated in a single case study in an educational setting, so their effectiveness in other settings will need further validation as well as by their use in the analysed literature.
- *Implementation Bias*: The use of JaCaMo (2022) facilitated the implementation of the design patterns. Although the results may not directly translate to other agent development frameworks, specially if they do not adhere to usual MAS practices and techniques.
- *Conclusion Validity*: The selected studies and the single running example may not provide sufficient evidence of the general applicability of the catalogue across DT domains. Therefore, our conclusions are limited to the feasibility of applying the catalogue in the studied contexts. Further case studies and industrial evaluations are needed to assess its broader applicability.
- *Reliability*: The qualitative coding process may be influenced by the interpretations of the authors. To improve the reliability, Section 4 describes the literature selection and coding procedures, and the extracted data and coding outcomes will be made available under request.

8 Conclusions and future work

With respect to **RQ1**, this study proposes a novel catalogue of agent design patterns aimed at supporting the implementation of the *reflection*, *memorization*, and *entanglement* properties of a DT, facilitating the development of a quality DT. The catalogue includes patterns for each DT property:

- **Reflection patterns:** Direct Reflection Design Pattern, Pipeline Design Pattern, and Fusion Design Pattern.
- **Memorization patterns:** Current-State Fact Design Pattern, Time-Series Data Design Pattern, Frame Model Design Pattern, and Ontology Model Design Pattern;
- **Entanglement patterns:** Indirect Observation Design Pattern, Continuous Direct Data Dispatcher Design Pattern, Event-Driven Direct Data Update Design Pattern, Periodic Direct Data Sending Design Pattern, and Action Dispatcher Design Pattern.

Regarding **RQ2**, different decision models are proposed to guide software architects in selecting the previous design patterns based on the *reflection* property (data sources and transformation needs), the *memorization* property (static and dynamic nature), and the *entanglement* property (connectivity, association and promptness). In addition, their application in the running example demonstrates that the proposed decision models support a

systematic selection of patterns. The implementation of these patterns confirms the feasibility of real-time monitoring, real-time decision-making, and the persistence of historical data. Moreover, the DT is capable of both modifying the environment and guiding the child's behaviour through messages. These capabilities reflect the quality of the developed DT satisfying the expected properties it should offer.

Finally, future work will focus on enhancing the applicability of the proposed catalogue of design patterns by extending it with new patterns associated with other DT properties, such as *predictability* or *composability*. Furthermore, this work adopts the definition of DT memorization proposed by Minerva et al. (2020), which characterizes it as the capability to store information about the physical twin, including its features, current and historical states, events, and contextual information. Future research should explore a broader conceptualization of DT memorization that also incorporates provenance information, modelling assumptions and paradata.

In parallel, because the *representativeness and contextualization* DT property is treated in this work as prerequisite specification concerns, future research should investigate requirements engineering methods to support them. In particular, techniques for systematically identifying which physical attributes should be represented in a DT would complement the proposed design patterns and help establish an end-to-end process that connects requirements elicitation with architectural design and implementation.

Moreover, although the catalogue highlights pattern-specific design concerns, it does not provide a formal method for identifying, prioritizing, negotiating, or evaluating conflicts among competing concerns such as fidelity, latency, storage cost, complexity, privacy, or safety. Addressing these conflicts remains an important avenue for future research. In this direction, we plan to define a quality model that explicitly relates and balances the DT properties addressed in this article. Such a model would enable a more systematic analysis of the tensions and complementarities among these properties, and could support pattern selection not only from an architectural perspective, but also with respect to DT quality attributes.

Another future work will be the extension of the existing tool for the MAS4DT methodology, which currently assists software architects in creating diagrams to identify relevant physical twin attributes, relationships, agents, and other elements required for the design of a DT based on MAS[1]. The tool will be enhanced to incorporate the catalogue proposed in this work, including the associated decision models. This extension will guide architects in selecting and combining the most appropriate agent design patterns according to the identified attributes, data sources, memorization requirements, and the required degree of entanglement between the physical twin and its DT.

A ISO/IEC 25010 quality characteristics used in the pattern catalogue

This appendix presents the definitions of the quality characteristics referenced throughout the pattern descriptions, particularly in the Forces, Benefits, and Drawbacks sections. To ensure consistency and a common understanding of these quality characteristics across all patterns, their definitions are based on the ISO/IEC 25010:2011, Systems and Software Engineering–Systems and Software Quality Requirements and Evaluation (SQuaRE)–System and Software Quality Models standard (25010, 2011).

[1] https://github.com/UCLM-LoUISE/PRJ2019-2gether-MAS4DT

- **Functional Suitability**: Degree to which a product or system provides functions that meet stated and implied needs when used under specified conditions.**Functional completeness**: Degree to which the set of functions covers all the specified tasks and user objectives.
- **Functional correctness**: Degree to which a product or system provides the correct results with the needed degree of precision.
- **Functional appropriateness**: Degree to which the functions facilitate the accomplishment of specified tasks and objectives.
- **Performance Efficiency**: Performance relative to the amount of resources used under stated conditions.**Time behaviour**: Degree to which the response and processing times and throughput rates of a product or system, when performing its functions, meet requirements.
- **Resource utilization**: Degree to which the amounts and types of resources used by a product or system when performing its functions meet requirements.
- **Compatibility**: Degree to which a product, system or component can exchange information with other products, systems or components, and/or perform its required functions, while sharing the same hardware or software environment.**Interoperability**: Degree to which two or more systems, products or components can exchange information and use the information that has been exchanged.
- **Reliability**: Degree to which a system, product or component performs specified functions under specified conditions for a specified period of time-**Fault tolerance**: Degree to which a system, product or component operates as intended despite the presence of hardware or software faults.
- **Availability**: Degree to which a system, product or component is operational and accessible when required for use.
- **Maintainability**: Degree of effectiveness and efficiency with which a product or system can be modified by the intended maintainers.**Modularity**: Degree to which a system or computer program is composed of discrete components such that a change to one component has minimal impact on other components.
- **Modifiability**: Degree to which a product or system can be effectively and efficiently modified without introducing defects or degrading existing product quality.
- **Security**: Degree to which a product or system protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization.**Integrity**: Degree to which a system, product or component prevents unauthorized access to, or modification of, computer programs or data.

Acknowledgements This paper has been partially funded by the R+D+i projects PID2022-140907OB-I00 and PID2025-171341OB-I00 funded by MICIU/AEI /10.13039/501100011033 and ERDF, EU. It has also been partially supported by Junta de Comunidades de Castilla-La Mancha/ERDF (SBPLY/24/180225/000020) and by the University of Castilla-La Mancha (2025-GRIN-38441). Elena Pretel holds a FPU21/02679 scholarship from Spanish Ministerio de Educación y Formación Profesional.

Author Contributions Elena Pretel: Conceptualization, Data curation, Methodology, Software, Visualization, Writing – original draft; Jos J GonzlezGarca: Methodology, Validation, Visualization, Writing – original draft; Elena Navarro: Conceptualization, Funding acquisition, Methodology, Writing – review & editing; Vctor LpezJaquero: Conceptualization, Funding acquisition, Methodology, Writing – review & editing; Schahram Dustdar: Validation, Writing – review & editing.

Funding This paper has been partially funded by the R+D+i projects PID2022-140907OB-I00 and PID2025-171341OB-I00 funded by MICIU/AEI /10.13039/501100011033 and ERDF, EU. It has also been partially supported by Junta de Comunidades de Castilla-La Mancha/ERDF (SBPLY/24/180225/000020) and by the University of Castilla-La Mancha (2025-GRIN-38441). Elena Pretel holds a FPU21/02679 scholarship from Spanish Ministerio de Educación y Formación Profesional.

Data Availability All data supporting the findings of this study can be made available on Zenodo upon request.

Declarations

Competing interests The authors declare no competing interests.

References

25010 I (2011) Systems and software engineering - systems and software quality requirements and evaluation (square) - system and software quality models. https://www.iso.org/obp/ui/#iso:std:iso-iec:25010:ed-1:v1:en

Cassar, A., Baker, D., & Ioannides, M. (2025). From digital twin to memory twin: a holistic framework for cultural heritage documentation, interpretation, and adaptive reuse. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XLVIII-M-9-2025:203–209. https://doi.org/10.5194/isprs-archives-XLVIII-M-9-2025-203-2025

Chella, A., Cossentino, M., Gaglio, S., et al. (2010). Agent-oriented software patterns for rapid and affordable robot programming. *Journal of Systems and Software, 83*, 557–57. https://doi.org/10.1016/j.jss.2009.10.035

Corkhill, D. (1991). *Blackboard systems. AI Expert pp* (pp. 41–47).

Croatti, A., Gabellini, M., Montagna, S., et al. (2020). On the integration of agents and digital twins in healthcare. *Journal of Medical Systems, 44*, 1–8. https://doi.org/10.1007/s10916-020-01623-5

Deugo, D., Oppacher, F., & Kuster, J. M. (1999). *Patterns as a means for intelligent software engineering*.

Dorri, A., Kanhere, S. S., & Jurdak, R. (2018). Multi-agent systems: A survey. *IEEE Access, 6*, 28573–28593. https://doi.org/10.1109/ACCESS.2018.2831228

Douglass, B. P. (2003). *Real-time design patterns: robust scalable architecture for real-time systems*. Boston: Addison Wesley.

Dupire, B., & Fernandez, E. B. (2001). *The command dispatcher pattern*.

Fernandez-Marquez, J. L., Arcos, J. L., Serugendo, G. D. M., et al. (2011). *Description and composition of bio-inspired design patterns: The gossip case* (pp. 87–96). IEEE. https://doi.org/10.1109/EASe.2011.15

Gamma, E., Helm, R., Johnson, R., et al. (1995). *Design Patterns: Elements of Reusable Object Oriented Software*. New York: Addison-Wesley.

Gayathri, R., & Uma, V. (2018). Ontology based knowledge representation technique, domain modeling languages and planners for robotic path planning: A survey. *ICT Express, 4*, 69–74. https://doi.org/10.1016/j.icte.2018.04.008

Glaser, B. G., & Strauss, A. L. (1999). *The Discovery of Grounded Theory: Strategies for Qualitative Research*. Routledge. https://doi.org/10.4324/9780203793206

González-García, J. J., Pretel, E., López-Jaquero, V., et al. (2025). Smart-pomodoro: Wearable technology for improving time-management, engagement and focus in children with ADHD. *International Journal of Human–Computer Interaction*, 1–31. https://doi.org/10.1080/10447318.2025.2542886

Gorodetsky, V. I., Kozhevnikov, S. S., Novichkov, D., et al. (2019). *The framework for designing autonomous cyber-physical multi-agent systems for adaptive resource management*. pp 52–64. https://doi.org/10.1007/978-3-030-27878-6_5

Grieves, M., & Vickers, J. (2017). *Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems*. https://doi.org/10.1007/978-3-319-38756-7_4.

Guo, J., & Lv, Z. (2022). Application of digital twins in multiple fields. *Multimedia Tools and Application*. https://doi.org/10.1007/s11042-022-12536-5

Heininger, R., & Stary, C. (2021). *Capturing autonomy in its multiple facets: A digital twin approach* (pp. 3–12). ACM. https://doi.org/10.1145/3445969.3450422

Hribernik, K., Cabri, G., Mandreoli, F., et al. (2021). Autonomous, context-aware, adaptive digital twins—state of the art and roadmap. *Computers in Industry, 133*, 103508. https://doi.org/10.1016/j.compind.2021.103508

Huang, J., Koroteev, D. D., & Rynkovskaya, M. (2023). Machine learning-based demand response in pv-based smart home considering energy management in digital twin. *Solar Energy, 252*, 8–19. https://doi.org/10.1016/j.solener.2023.01.044

IDTA. (2023). *Part 3a: Data specification-iec 61360 specification of the asset administration shell*. https://www.industrialdigitaltwin.org/

ISO. (2021). *Iso 23247, automation systems and integration - digital twin framework for manufacturing*.

JaCaMo. (2022). *Jacamo*. https://jacamo-lang.github.io/

Khail, W. S., & Schnizer, P. (2024). *Patterns in digital twin development* (pp. 1–8). ACM. https://doi.org/10.1145/3698322.3698325

Kitchenham, B., Brereton, O. P., Budgen, D., et al. (2009). Systematic literature reviews in software engineering – a systematic literature review. *Information and Software Technology, 51*, 7–15. https://doi.org/10.1016/j.infsof.2008.09.009

Korenhof, P., Blok, V., & Kloppenburg, S. (2021). Steering representations—towards a critical understanding of digital twins. *Philosophy & Technology, 34*, 1751–1773. https://doi.org/10.1007/s13347-021-00484-1

Kreuzer, T., Papapetrou, P., & Zdravkovic, J. (2024). Artificial intelligence in digital twins—a systematic literature review. *Data & Knowledge Engineering, 151*, 102304. https://doi.org/10.1016/j.datak.2024.102304

Kritzinger, W., Karner, M., Traar, G., et al. (2018). Digital twin in manufacturing: A categorical literature review and classification. *IFAC-PapersOnLine, 51*, 1016–1022. https://doi.org/10.1016/j.ifacol.2018.08.474

Lektauers, A., Bikovska, J., & Bolsakovs, V. (2022). *An agent-directed digital twin framework for simulation-based training* (pp. 1–6). IEEE. https://doi.org/10.1109/ITMS56974.2022.9937103

Liu, W., Li, B., Xie, W., et al. (2023). Energy efficient computation offloading in aerial edge networks with multi-agent cooperation. *IEEE Transactions on Wireless Communications*, 1–1. https://doi.org/10.1109/twc.2023.3235997

Liu, Y., Lo, S. K., Lu, Q., et al. (2025). Agent design pattern catalogue: A collection of architectural patterns for foundation model based agents. *Journal of Systems and Software, 220*, 112278. https://doi.org/10.1016/j.jss.2024.112278

Li, Y., & Zhang, W. (2023). Traffic flow digital twin generation for highway scenario based on radar-camera paired fusion. *Scientific Reports, 13*, 642. https://doi.org/10.1038/s41598-023-27696-z

Loh, H. W., Ooi, C. P., Barua, P. D., et al. (2022). Automated detection of ADHD: Current trends and future perspective. *Computers in Biology and Medicine, 146*, 105525. https://doi.org/10.1016/j.compbiomed.2022.105525

Luder, A., Cala, A., Zawisza, J., et al. (2017). *Design pattern for agent based production system control — a survey* (pp. 717–722). IEEE. https://doi.org/10.1109/COASE.2017.8256187

Lu, Y., Xie, J., Safdar, M., et al. (2024). *An overarching quality evaluation framework for additive manufacturing digital twin* (pp. 676–682). IEEE. https://doi.org/10.1109/CASE59546.2024.10711607

Macías, A., Navarro, E., & Cuesta, C. E. (2026). Requirements engineering for digital twins: a cross-domain systematic literature review. In *18th International Conference on the Quality of Information and Communications Technology*. https://doi.org/10.1007/978-3-032-20631-2_9

Macías, A., Muñoz, D., Navarro, E., et al. (2024). Data fabric and digital twins: An integrated approach for data fusion design and evaluation of pervasive systems. *Information Fusion, 103*, 102139. https://doi.org/10.1016/j.inffus.2023.102139

Mammadova, M. H., & Ahmadova, A. A. (2023). Development of digital twin ecosystem and ontology in medicine. *Technology Transfer: Fundamental Principles and Innovative Technical Solutions*, 21–23. https://doi.org/10.21303/2585-6847.2023.003203

Marks, P., Muller, T., Vogeli, D., et al. (2018). *Agent design patterns for assistance systems in various domains - a survey* (pp. 168–173). IEEE. https://doi.org/10.1109/COASE.2018.8560391

Mehdi, H. (2025). *Frame-based agents: Structural mechanisms for cognition*.

Meszaros, G. (1997). *Archi-patterns 1: A process pattern language for defining architectures*.

Minerva, R., Lee, G. M., & Crespi, N. (2020). Digital twin in the iot context: A survey on technical features, scenarios, and architectural models. *Proceedings of IEEE, 108*, 1785–1824. https://doi.org/10.1109/JPROC.2020.2998530

Mohamad, R., Deris, S., & Ammar, H. (2006). *Agent design patterns framework for mase/poad methodology* (pp. 521–52). IEEE. https://doi.org/10.1109/AICCSA.2006.205140

MongoDB. (2026). Mongodb manual. https://www.mongodb.com/docs/, accessed: 25 Mar 2026

Oluyomi, A., Karunasekera, S., & Sterling, L. (2004). *An agent design pattern classification scheme: Capturing the notions of agency in agent design patterns* (pp. 456–463). IEEE. https://doi.org/10.1109/APSEC.2004.19

Pan, J., Cranefield, S., & Carter, D. (2003). *A lightweight ontology repository* (pp. 632–63). ACM. https://doi.org/10.1145/860575.860677

Pretel, E., Moya, A., Navarro, E., et al. (2024). Analysing the synergies between multi-agent systems and digital twins: A systematic literature review. *Information and Software Technology, 174*, 107503. https://doi.org/10.1016/j.infsof.2024.107503

Pretel, E., Zhinin-Vera, L., Navarro, E., et al. (2025). Mas4dt: A novel proposal for developing digital twins following a multi-agent system approach. *Journal of Systems and Software*, 112344. https://doi.org/10.1016/j.jss.2025.112344

Raghunandan, A., Kalasapura, D., & Caesar, M. (2023). *Digital twinning for microservice architectures* (pp. 3018–302). IEEE. https://doi.org/10.1109/ICC45041.2023.10279802

Rudakov, V., Timur, M., & Yedilkhan, A. (2023). *Comparison of time series databases* (pp. 1–4). IEEE. https://doi.org/10.1109/ICECCO58239.2023.10147153

Schmidt, D. C., Stal, M., Rohnert, H., et al. (2013). *Pattern-Oriented Software Architecture*. Patterns for Concurrent and Networked Objects: Wiley.

Skobelev, P., Tabachinskiy, A., Simonova, E., et al. (2021). *Digital twin of rice as a decision-making service for precise farming, based on environmental datasets from the fields* (pp. 1–8). IEEE. https://doi.org/10.1109/ITNT52450.2021.9649038

Tao, F., Zhang, M., Liu, Y., et al. (2018). Digital twin driven prognostics and health management for complex equipment. *CIRP Annals, 67*, 169–172. https://doi.org/10.1016/j.cirp.2018.04.055

Tekinerdogan, B., & Verdouw, C. (2020). Systems architecture design pattern catalog for developing digital twins. *Sensors, 20*, 510. https://doi.org/10.3390/s20185103

Vermeulen, A., Beged-Dov, G., & Thompson, P. (1995). The pipeline design pattern. In *Proceedings of OOPSLA'95 Workshop on Design Patterns for Concurrent, Parallel, and Distributed Object-Oriented Systems*. Citeseer.

Wedyan, F., & Abufakher, S. (2020). Impact of design patterns on software quality: a systematic literature review. *IET Software, 14*, 1–17. https://doi.org/10.1049/iet-sen.2018.5446

Xu, H., Wu, J., Pan, Q., et al. (2023). A survey on digital twin for industrial internet of things: Applications, technologies and tools. *IEEE Communications Surveys and Tutorials, 25*, 2569–2598. https://doi.org/10.1109/COMST.2023.3297395

Yuan, J., Sun, M., Xie, J., et al. (2022). A multi-agent system construction method for substation digital twin. In *2022 2nd International Conference on Electrical Engineering and Mechatronics Technology (ICEEMT)* (pp. 397–402). https://doi.org/10.1109/ICEEMT56362.2022.9862764

Yun, T. S., Moon, I. C., & Kim, Y. C. (2020). *Agent-based modeling and simulation on residential population movement patterns: The case of sejong city* (pp. 620–63). IEEE. https://doi.org/10.1109/WSC48552.2020.9383874

Zhang, H., Qi, Q., & Tao, F. (2022). A consistency evaluation method for digital twin models. *Journal of Manufacturing Systems, 65*, 158–168. https://doi.org/10.1016/j.jmsy.2022.09.006

Zhou, X., Jin, Y., Zhang, H., et al. (2016). *A map of threats to validity of systematic literature reviews in software engineering* (pp. 153–160). IEEE. https://doi.org/10.1109/APSEC.2016.031

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.



Elena Pretel Ph.D. in Advanced Computer Science (2025, University of Castilla-La Mancha, UCLM), MSc (2022, UCLM), and BSc in Computer Science (2021, UCLM), is a postdoctoral researcher at UCLM. She is currently a member of the "Laboratory of User Interaction and Software Engineering" (LoUISE) research group at the "Instituto de Investigación en Informática de Albacete" (I3A), Spain, and has completed a research stay at TU Wien, Austria. Her research focuses on Digital Twins, Multi-Agent Systems, and Artificial Intelligence for the development of autonomous decision-making systems.



José J González-García MSc (2024, UCLM), PhD Student in Computer Science. He holds a research position in the research group LoUISE. His current research focuses on the application of wearable technology into the fields of Digital Twins, serious games, behavioral ADHD treatment, and social environments analytics.



Elena Navarro is a full professor at the Computing Systems Department of the UCLM (Spain) where she lectures in Software Engineering. Prior to this position, she did research at the Computer Science Laboratory of the AUA (Greece) funded by the EU TMR Program. She had previously served on the staff of the Regional Government of Murcia. She is author of more than 120 papers, being more than 50 papers in relevant journals indexed in JCR, regarding software architecture and requirements engineering. She is involved in numerous R&D projects as well as in Knowledge transfer projects. Her current research interests are related to the design and development of Digital Twins with a special focus on Software Architecture and Requirements Engineering.



Víctor López-Jaquero PhD (2005, UCLM), is a Full Professor at University of Castilla-La Mancha, Spain. His research interests include human-computer interaction, human digital twins, multi-agent systems, model-driven development and e-health. He has been involved in research projects regarding those topics for the last 25 years. He has authored more than 100 papers in relevant conferences and journals.



Schahram Dustdar is a Full Professor of Computer Science at TU Wien, where he leads the Distributed Systems Group, and he is also affiliated as an ICREA research professor at Universitat Pompeu Fabra (UPF) in Barcelona (https://dustdar.prof). He is widely known for pushing the frontier of elastic, dependable cloud-to-edge systems and the Computing Continuum, turning cutting-edge research into practical foundations for modern distributed intelligence. An IEEE Fellow, ACM Distinguished Scientist/Speaker, and member of Academia Europaea, he's a sought-after keynote speaker recognized for shaping how large-scale systems adapt, scale, and stay resilient in the real world.

Authors and Affiliations

Elena Pretel[1] · José J González-García[1] · Elena Navarro[1] · Víctor López-Jaquero[1] · Schahram Dustdar[2,3]

✉ Elena Navarro
elena.navarro@uclm.es

Elena Pretel
mariaelena.pretel@uclm.es

José J González-García
josejesus.gonzalez@uclm.es

Víctor López-Jaquero
victormanuel.lopez@uclm.es

Schahram Dustdar
dustdar@dsg.tuwien.ac.at

1 LoUISE Research Group, I3A, University of Castilla-La Mancha, Albacete 02071, Spain

2 TU Wien, Vienna 1040, Austria

3 UPF ICREA, Barcelona 08018, Spain