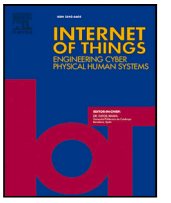
Internet of Things 31 (2025) 101560

Contents lists available at ScienceDirect

Internet of Things

journal homepage: www.elsevier.com/locate/iot



Research article

Distributing intelligent functionalities in the Internet of Things with agents and Digital Twins

Samuele Burattini [a], Stefano Mariani [b,*], Sara Montagna [c], Marco Picone [b], Alessandro Ricci [a]

[a] Università degli Studi di Bologna, Cesena, Italy
[b] Università di Modena e Reggio Emilia, Reggio Emilia, Italy
[c] Università di Urbino, Urbino, Italy

ARTICLE INFO

Keywords:
Distributed intelligence
Autonomous agents
Digital Twins
Internet of Things
Software engineering

ABSTRACT

Autonomous Agents (AAs) and Digital Twins (DTs) are two widely used abstractions in the literature about the engineering of "intelligent" Internet of Things (IoT) systems and applications. However, their role can be considered partially overlapping given the fragmented landscape of approaches emerging from the literature. There, in fact, either AAs or DTs, or their combination, are sometimes used for achieving the same goals. In this paper, we attempt to clarify similarities and differences of these abstractions and argue that the choice to use AAs or DTs (or an integration of the two) should stem from a principled analysis of the IoT system requirements. That is, by matching the desired *intelligent functionalities* with the properties of the two abstractions to find the most appropriate one. Accordingly, we *(i)* analyse the state-of-the-art approaches to identify how AAs and DTs are currently used to encapsulate and distribute intelligent functionalities across IoT system components; *(ii)* propose a set of *principles* to assist designers in choosing the most suitable abstraction for a given functionality; and *(iii)* discuss exemplary architectures that may arise from applying such principles. To conceptually validate our contribution, we analyse the practical case of an intelligent manufacturing system and show how following the outlined principles leads to interesting properties in the final system design.

1. Introduction

The Internet of Things (IoT) paved the way for a new breed of complex software systems that are *(i)* deeply intertwined with the physical world, *(ii)* made up of several "intelligent" subsystems distributed both *conceptually*, in terms of abstractions, and *physically*, in terms of deployments, and *(iii)* requiring an increasing amount of intelligent functionalities (e.g. prediction, adaptive control, what-if simulation) for autonomous decision-making [1,2]. Such functionalities may rely on different information models, techniques, and technologies (e.g. numerical control software vs. machine learning models).

Autonomous Agents (AAs) and Digital Twins (DTs) are gaining increasing attention in the literature about the distribution of intelligence in IoT systems and applications [3–5]. AAs encapsulate reasoning routines and decision-making criteria tailored to a given goal. Hence, they naturally lend themselves to implement the perception-action feedback loop typically required in the IoT to realise the "actionable knowledge" paradigm [6]. DTs digitalise an entity of interest in the physical world to decouple applications and services from the technical intricacies of interacting with such entities. Also, they may offer additional digital

* Corresponding author.
E-mail addresses: samuele.burattini@unibo.it (S. Burattini), stefano.mariani@unimore.it (S. Mariani), sara.montagna@uniurb.it (S. Montagna), marco.picone@unimore.it (M. Picone), a.ricci@unibo.it (A. Ricci).

https://doi.org/10.1016/j.iot.2025.101560
Received 29 August 2024; Received in revised form 25 February 2025; Accepted 26 February 2025
Available online 8 March 2025

functions (i.e. *augmenting* the behaviour of physical entities [7]). Thus, they both naturally fit the IoT context in bridging together the "cyber" and the "physical" parts of a system. Consequently, to date, several approaches and architectures have been proposed employing either one abstraction or the other, or even both combined [8–10], to enable *distributed intelligence* [1,11].

However, it is not always clear what the *relationship* between these abstractions is at both the architectural and technological level: namely, whether AAs and DTs are interchangeable, (partially) overlapping, alternatives, complementary, or anything else. Some works exploit exclusively AAs to encapsulate intelligent functionalities or even only to mediate access to resources [12,13]. Some others instead rely solely on DTs and put intelligent functions therein [14,15]. In those works that propose an integration, it happens in many different ad-hoc ways: for instance, with "multi-agent based DTs" to digitalise complex systems of the real world (such as a whole smart city [16,17]), or with AAs using DTs as a digital substrate to deal with interoperability [18]. As a consequence, we observe a problem called, in software engineering terms, *fragmentation*: the existence and proliferation of several solutions to the same problems, which have no clear relationships with each other (e.g. are they fundamentally different or simply exploit different technologies?), and that are conceived for specific applications. Therefore, they are typically hard to generalise and cause "reinventing the wheel" multiple times, virtually for any different application and/or deployment scenario.

In fact, as most of these approaches are proposed in the context of a specific application or intended goal, there is little interest in providing IoT systems and application engineers with general principles to guide their design choices regardless of the specific application domain and/or goal. We argue that this lack of clarity is problematic for the broader IoT community, as it leads to a variety of approaches that may tackle similar problems with different solutions, keeping closed vertical silos between partially overlapping communities. Additionally, resorting to custom integrations between AAs and DTs may hinder the development, maintenance, and evolution of interconnected systems developed independently.

To provide a starting common ground for these integrated approaches to emerge, and create bridges between the AAs and DTs communities, in this paper we attempt to sort this lack of clarity out from a *software engineering* standpoint. That is, we adopt the perspective of the designer of an IoT system or application that wants to understand how to distribute intelligence across system components while employing AAs and DTs. The motivation to adopt such a standpoint lies in the increasingly prominent role that software abstractions, paradigms, and architectures have in IoT systems engineering [19,20]. For doing so, we suggest that the choice to use either one, the other, or a combination of the two abstractions should stem from a principled analysis of the requirements of the desired functionalities of the intelligent IoT system. Furthermore, based on general-purpose, domain- and technology-agnostic design principles, we discuss exemplary IoT *micro*-architectures that may arise depending on the system at hand. Such architectures, although different, are not in contrast with one another, nor are they intended to be "reference" architectures for a given domain. On the contrary, we shift the focus to the principles motivating why a given architecture, or a combination thereof, can be suited for a specific use case.

Accordingly, our contribution unfolds as follows:

- we analyse the literature about using AAs or DTs for the engineering of intelligent IoT systems, to derive common intelligent functionalities in a bottom-up fashion— to avoid the endless debate of defining *intelligence* in the first place;
- we propose a set of design principles (revolving around the fundamental software engineering principle of *separation of concerns*) to assist designers in choosing the most suitable abstraction for a given functionality, by analysing its requirements in an application-, domain-, and technology-agnostic way;
- we discuss exemplary micro-architectures (μ-arch, for short) that may arise from the application of such principles, both in general and for specific edge cases;
- we conceptually validate how such principles and μ-archs can assist designers in deriving an adequate conceptual architecture for their intelligent functionalities at hand, in the practical case of an intelligent manufacturing application.

The rest of the paper is thus organised as follows: Section 2 recalls the basics of AAs and DTs, and surveys their utilisation in the domain of the IoT; Section 3 presents the design principles we propose, illustrating how they help in choosing AAs or DTs for a given functionality, and discussing how AAs and DTs can be used as complementary architectural abstractions; Section 4 conceptually validates the proposal following the analysis of a practical case study using the aforementioned principles and architectural choices; Section 5 enumerates several open challenges to be dealt with for the effective integration of AAs and DTs in the engineering of intelligent IoT Systems; finally, Section 6 wraps up with the concluding remarks.

2. Autonomous agents and Digital Twins in the IoT

In this section, we frame our proposal in the context of Distributed Intelligence (DI), with particular reference to the IoT domain. To do so, we recall the definitions and main characteristics of AAs and DTs, and observe how these abstractions are used to bring intelligence within IoT systems through an analysis of the literature. Doing so serves to identify the typical intelligent functionalities that are considered in the literature (e.g. prediction, time series forecast, fault diagnosis, etc.), to better frame the heterogeneity of intelligent subsystems that exist in IoT systems.

2.1. Distribution of Intelligence in the iot

The area of *Distributed Intelligence* (DI) [21–23] broadly addresses the challenges of both *(i)* combining different AI approaches to process data streams, perform reasoning, and automate decision-making, and *(ii)* distributing them across the Computing Continuum [24] within an IoT system. The latter challenge includes both *physical* distribution (i.e. on which processing device the software is going to be deployed) and *conceptual* distribution (i.e. which logical component of the system will be assigned the responsibility for a certain intelligent function). Physical distribution is often driven by computational requirements such as latency and computational power. In fact, the concept of Computing Continuum emerged from the need of stratification of IoT systems' infrastructure [25]. A recent survey [11] frames the efforts to deploy intelligence in this Computing Continuum.

Our focus shifts to the conceptual distribution, a more nuanced issue due to the lack of equally strong guiding principles or constraints. In this context, engineers and developers must rely on best practices or general principles of software engineering, such as separation of concerns, loose coupling, and high cohesion, among others [26]. A significant challenge here is how to *encapsulate* intelligent functionalities into reusable software components that can *(i)* monitor the system and infer situations, *(ii)* act *autonomously* and efficiently, and *(iii)* accurately encapsulate the decision-making criteria leading from acquired information (as perceived from the IoT system) to action plans. Given that these functionalities may depend on various information sources, can be implemented with different algorithms (ranging from machine learning and reinforcement learning to logical inference), and can be provided in numerous ways today, it is essential to identify clear guiding principles and first-class abstractions that IoT system designers can use to effectively build and distribute intelligence. Then, suitable architectures and methodologies can be derived, flexible enough to adapt to the different specificities of any application scenario. As AAs and DTs are increasingly recognised in the literature as reference conceptual tools to encapsulate and distribute intelligence across an IoT system [17,27,28], we will frame our proposal around these two abstractions.

2.2. Autonomous agents

Autonomous (software) agents (AAs) can be concisely defined as computational entities that encapsulate the criteria for undertaking decision-making (*autonomy*) toward the achievement of a given goal. They do so based on available information regarding the external context they can perceive and act upon—they are *situated* in their *environment*. The concept gained popularity in the early 1990s [29] and became central in Artificial Intelligence as the main abstraction adopted to engineer intelligent and adaptive software systems [30–32].

Common architectures for implementing software agents reported in the literature include planning, logical, and learning agents [33–35]. These architectures share the ability to reason based on a set of environmental perceptions. To achieve their tasks they can also seamlessly cooperate with other agents in a Multi-Agent System (MAS) [36], where interaction among agents with direct or indirect communication is key to providing the overall intelligent functionalities they are designed for. It also helps to break down a complex system into individual components. For these reasons the architecture of an AA is usually focussed on supporting the AA reasoning and decision making workflow efficiently, and on providing to the programmer the right set of abstractions at the right level of abstraction for the human mind, regardless of infrastructural aspects. For instance, a common architecture that has shown to be extremely powerful in representing heterogeneous kinds of reasoning is the *cognitive architecture* [37]. There, the AA is able to integrate, to some extent, control-driven behaviours explicitly programmed with data-driven ones acquired via machine learning techniques. For an overview of the kind of AA architectures the interested reader is forwarded to Ref. [38]. In principle, the architecture of an AA only imposes minimal requirements on the underlying execution infrastructure, and should enable the definition of modular components (the AAs themselves) that are focussed on carry out a given goal (or only a few related ones at most). It has been shown in fact that better than letting an AA complexity grow out of hand uncontrollably, a MAS is usually to be preferred to decompose the problem to be dealt with in "smaller" problems of more manageable complexity [39].

Agents and MAS have thus been widely used in many domains primarily as a context-aware model and technology to encapsulate intelligent behaviour in *distributed* and *reusable* software components. This includes the IoT domain, that is a good fit for agents' situated decision-making abilities [3,40]: sensors and actuators provide the means to perceive and affect the real world (a physical environment), while agents encapsulate the system's *goals* and the reasoning processes needed to bring the system in the desired state of affairs—by closing the feedback loop between sensing and acting. From a distributed system perspective, MAS improves IoT by addressing autonomy and heterogeneity, enabling device and data discovery through semantic service descriptions, fostering trust through reputation and incentives, and facilitating collective intelligence [41].

Looking at the specific literature that exploits AAs (and MAS) in the IoT, the following emerge as the main intelligent functionalities they encapsulate (summarised in Table 1).

- *Logic inference* [42,43],[45], broadly intended as the task of executing well-defined reasoning steps to synthesise novel information or action plans [44], based on some logic formalism (e.g. first-order logic).
- *Adaptive control* [12,13,46],[44,47,48], that is, the ability to quickly react to (possibly) unexpected events to reconfigure the system itself, either to provide the same functionalities despite the disruption or to opportunistically improve its functionalities given the chance (e.g. elastically scale up when new computational resources are made available [41]).
- *Learning of control policies* [49,50],[51–54] from available information and execution context to adapt to dynamic operational environments. *Reinforcement learning* [57] agents are a notable example of gaining more traction and success stories, but not necessarily the only form of learning that can be used. Unlike adaptive control, here the adaptation is learnt in a data-driven way, not programmed.

Table 1
Selection of literature on AAs exploitation in IoT for delivering different intelligent functionalities.

Intelligent functionality	Representative papers	Notes
Logic inference	[42–45]	Works applying logic-based approaches in IoT systems to improve explainability and deductive reasoning of the system components.
Adaptive control	[12,13,44,46–48]	Works showcasing the effectiveness of adaptive behaviour of agents in IoT settings, highlighting properties such as context- and self-awareness as well as planning for dynamic reconfiguration.
Learning of control policies	[49–54]	Works showcasing applied scenarios of Multi-Agent Reinforcement Learning and Deep Reinforcement Learning to learn cooperation policies.
Simulation	[44,55,56]	Works describing simulations of IoT scenarios performed with a multi-agent based model to simulate the interactions between autonomous components.

- *Simulation* [55,56],[44] of complex processes that require sophisticated modelling or multiple interacting entities (as in Multiagent-based Simulation). Here, agents are usually used to encapsulate the expert domain knowledge and interaction patterns required to truthfully represent and reproduce the properties and behaviour of the simulated system, and allow manipulation of it in a virtual environment.

2.3. Digital Twins

Digital Twins (DTs), initially introduced in the 2010s within the context of product lifecycle management to enhance tracking in manufacturing settings, have significantly evolved. Originally focused on improving monitoring and decision-making in industrial domains, DTs have expanded their scope in recent years, driven by advances in key technologies such as the Internet of Things (IoT), machine learning, cloud and edge computing, and big data analytics [58]. The rapid development of DTs is a concern in different communities, as the concept may (possibly partially) overlap with existing methods and tools [59]. Nevertheless, its large-scale adoption is proving effective in delivering the idea of enhancing very different application domains through the creation of comprehensive software models of the physical world [60]. As the concept of DT is in continuous evolution and may be very different depending on the research communities and application domains, for the scope of this paper, we adopt the following DT definition taken from [7]:

> A DT is a comprehensive software representation of an individual physical object. It includes the properties, conditions, and behaviour (s) of the real-life object through models and data. A DT is a set of realistic models that can simulate the behaviour of an object in the deployed environment. The DT represents and reflects its physical twin and remains its virtual counterpart throughout the entire lifecycle of the object.

This definition highlights the functional role that DTs have in an IoT system. DTs serve as essential tools to bridge the gap between physical assets and cyberspace, digitalise the state of the real system, and enable continuous monitoring, analysis, and optimisation. This interaction is inherently *bidirectional*, allowing DTs to not only observe but also control and interact with their physical counterpart [7,18,61].

DTs are rapidly transcending their industrial origins, emerging as a universal framework for bridging the physical and digital worlds [60,62]. They offer robust conceptual and technological foundations for digitalising strategic physical assets within organisations, including products, machines, buildings, users, and operational processes. Simultaneously, the widespread adoption of IoT technologies and standards is broadening the applicability of DTs, enabling their integration beyond traditional vertical applications into comprehensive enterprise ecosystems. A key driver behind this surge in interest lies in the symbiotic relationship between IoT and DTs. On the one hand, IoT technologies are fundamental in enabling the existence of DTs [63], since "things" must connect seamlessly with software components to form synchronised digital counterparts. On the other hand, DTs significantly enhance the capabilities of IoT devices by expanding the range of functions and behaviours they can deliver [64]. This is evidenced by the increasing focus of standardisation bodies [65–67] and the active participation of various consortiums in the definition of the frameworks and guidelines that will shape the future of DT ecosystems.

Looking forward, these advancements point toward a future characterised by the pervasive transformation of the physical world into software-driven ecosystems [68]. This vision entails highly dynamic, interconnected, and interoperable DTs spanning various application domains and functioning seamlessly across different network layers, from cloud infrastructures to edge environments [69,70]. Such ecosystems promise unprecedented capabilities for monitoring, analysis, and control, redefining how physical assets interact with the digital realm.

Table 2
Selection of literature on DTs exploitation in IoT for delivering different intelligent functionalities.

Intelligent functionality	Representative papers	Notes
Prediction and learning	[15,76–82]	Prediction of future states of a PA is one of the most common intelligent functionalities delivered by DTs, together with simulation (see below). Such predictions are increasingly delivered via data-driven, machine learning techniques.
Inference	[83–86]	DTs are also used to synthesise novel information either based on historical data gathered about the PA, or by aggregating the data of multiple PAs that are so tightly coupled in the physical world to be worth digitalising them altogether with a single DT.
Simulation	[14,85,87–92]	Simulation of alternate configurations or working conditions for a given PA is another typical application of DTs.

While DTs are emerging as a reference approach for bridging the physical and digital worlds, alternative approaches are also available. For example, the Asset Administration Shell (AAS) in the industrial domain offers a standardised digital representation of assets, focussing on information management and integration. However, unlike DTs, the AAS is less concerned with models, behaviours, and the creation of digital replicas with augmented or potentially intelligent capabilities. Nevertheless, AAS and DTs can collaborate to take advantage of the benefits of digitalisation, modelling capabilities, and lifecycle synchronisation [71]. Another example is in the networking domain, where softwarization enabled flexible infrastructure management, yet the community is increasingly adopting DTs to address the growing complexity of interconnected and distributed network systems [72]. DTs go beyond traditional software solutions by providing unified, interoperable layers that seamlessly integrate into larger cyber–physical ecosystems.

While domain-specific software solutions can simplify interactions with physical systems, they often introduce increased architectural complexity and limited interoperability. This makes them less scalable and adaptable across diverse systems, particularly when considering the challenges and intricacies of structured and distributed cyber–physical systems. In contrast, DTs offer a distinctive advantage by providing a unified cyber–physical layer that decouples physical complexity from digital operations. This approach allows bringing intelligence closer to the point where data are generated and acted upon, fostering collaboration across domains and organisations while mitigating the risk of fragmented, domain-specific solutions [73,74].

Another concept closely related to that of DT, that somewhat extends it, has been proposed in [75] in the form of a *Digital Triad*: a *lifecycle* twin is added to the digital and physical twins. The goal of this new component is that of encapsulating the "engineering design knowledge" (quoting from the cited paper) necessary to evolve the other two models in a way that is (quoting again) "tailored to a specific application of digital manufacturing" and "customized based on the functional requirements of digital manufacturing". Although this addition goes in the direction of encapsulating intelligence and decision making into DTs, we argue (especially in Section 3) that it appears to *reinvent the wheel* with respect to the advances happened in AAs and multi-agent systems research in the last decades—as introduced in Section 2.2. Thus, in contrast, we propose to re-use AAs to distribute intelligent functionalities among them and DTs, while also providing specific design guidelines for doing so, in a way that is "customized based on the functional requirements of digital manufacturing"—hence, in line with the view put forward in [75].

Among the key properties that are considered for DTs in IoT [7] there are: *(i) representativeness*, referring to the degree to which a DT accurately reflects its physical counterpart in terms of properties, actions, events, and relationships; *(ii) contextualisation*, that entails selecting and representing only the relevant features, properties, and information of the PA that are essential for achieving the intended purpose (digitalising the PA); *(iii) augmentation*, as the possibility of introducing new software functionalities into the DT that would not necessarily have been supported directly by the PA; and *(iv) servitisation*, putting the focus on functionalities that are offered by the DT *as-a-service* for consumer applications.

The use of DTs now extends beyond manufacturing to encompass several domains such as smart buildings [93], agriculture [94], and healthcare [95], all with the common denominator of making extensive usage of IoT deployments as key enabler. There, DTs have been used mostly to encapsulate the following intelligent functionalities (summarised by Table 2).

- *Prediction* [15,76–79],[80–82], that is, predicting future states or behaviours for the PA of the IoT infrastructure digitalised by the DT. Such a prediction happens not only through accurate modelling of the PA dynamics but also in a data-driven way via *machine learning*.
- *Inference* [83,84],[85,86], that is, deriving novel or hidden information from the analysis and/or aggregation of existing one. Contrary to the case of AAs' *logic*, symbolic inference, DTs deliver inference via data-driven methods, mostly.
- *Simulation* [14,87–90],[85,91,92] of alternate configurations, inputs, outputs, or working conditions for the PAs associated with DTs. Simulation is often used for what-if analysis of the IoT system behaviour and fault diagnosis.

Table 3
Mapping of design principles to the target use case and broad categories of intelligent functionalities (Specif. = Specificity, Abst. = Abstraction, n.a. = not applicable, pure digitalisation function).

Overall objective	Functionalities	Kind	Specif.	Scoping	Timing	Abst.	μ-arch
Energy monitoring	Data collection	n.a.	General	Local	Explicit	DT	
	Anomaly Detection	Inference/ Prediction	General	Local	Explicit	DT	E
	Corrective Reaction	Inference/ Planning	Specific	Local	Implicit	Agent	
Intelligent coordination	Activity Monitoring	Inference/ Prediction	General	Local	Explicit	DT	
	Dynamic Planning	Planning	Specific	Global	Explicit	Agent	A
	Task Delegation	n.a.	General	Local	Implicit	DT	
Production optimisation	Decision-making	Inference	Specific	Global	Explicit	Agent	
	Process Monitoring	Inference	General	Global	Explicit	DT	A
	Task Delegation	n.a.	General	Local	Implicit	DT	

2.4. Synergistic usage of AAs and DTs

AAs and DTs are not only used separately to deliver intelligent functionalities in a mutually exclusive way, or as alternative solutions to overlapping requirements. Their synergistic usage is already seen in the available literature, especially in industrial deployments. Here, we briefly report recent surveys that highlight the rich intersection of these research areas.

The contamination of AAs in DTs is mostly related to the need to adopt a variety of artificial intelligence techniques in IoT applications, both to enhance the modelling capabilities of complex assets and to implement automatic controls. Minerva et al. present a classification of DTs in relation to the level of intelligent behaviour the DT exhibits [96]. They identify the ultimate level of intelligent DT as a proactive (or autonomic) DT, capable of enacting autonomous behaviour based on the physical counterpart's current or future context. This has a clear link with AAs, that are primarily used to model and encapsulate autonomous and intelligent behaviour in software systems.

Authors of [17] deliver a systematic literature review considering both MAS to create DTs and MAS to exploit DTs. Table 2 therein highlights how manufacturing is the main use case for synergistic AAs and DTs usage. This suggests that Industry 4.0 enabled by IoT technologies is a driving force for AAs and DTs applications. Furthermore, Table 3 therein emphasises that 73% of the surveyed papers do not disclose the AAs and DTs development framework and that almost 8% propose their own (second largest percentage behind a "team" of 4 agent-oriented development frameworks). This suggests a great fragmentation of uncoordinated proposals, generating "reinvention of the wheel", especially with respect to the integration of AAs and DTs.

The survey in [97], instead, has its main focus on DTs considering whether and how AAs (especially MAS) are used to complement the functionalities of DTs. Of particular relevance for the present article is Fig. 5 therein, which reports the vision of an "Intelligent Product" (IP) from the literature. Such IP fosters the synergistic usage of DTs and AAs. DTs help create an "intelligent being", whereas AAs augment it with an "intelligent agent" to make it an IP. The ARTI reference architecture is inspired by these concepts, and it has been among the first architectures proposed to consider the combined usage of AAs and DTs (and the most widely adopted). Finally, it is relevant that also here manufacturing and IoT emerge as the main application domains for the integrated usage of AAs and DTs.

The last survey we mention is [98], where the focus is on a one-way integration of AAs and DTs: AAs supporting DTs implementation. A practical example of such an integration is given in [9], where AAs are used to create a complex DT endowed with "intelligent" functionalities (e.g. prediction). An interesting takeaway that the authors of [98] get from the surveyed literature is that a main driving force behind this specific integration is exactly endowment of DTs with AA properties. In Section 3 we will see how this need is captured by our principles and proposed micro-architectures for AAs and DTs integration.

Providing the complete overview of all the available AAs and DTs-based architectures for IoT scenarios is out of the scope of this paper—the interested reader is referred to these surveys. However, it may be beneficial for the reader to report some selected examples. In [99], for instance, an AA is used to aggregate information from different DTs in a manufacturing scenario, to predict faults in machinery and (re-)optimise production scheduling. Another example in manufacturing is in [100]. There DTs are used mainly as an abstraction layer over the whole manufacturing system, providing a uniform access layer to AAs. AAs are used mainly to support automatic feedback control (digital to physical) and communication and coordination between robot AAs, task AAs, and workstation AAs. In [16], instead, AAs are used to build the complex DT of a whole city.

It is also worth noting that there may be literature, such as Ref. [75], where the term "autonomous agent" does not appear, but whose proposal is well aligned with the literature about AAs. For instance, in the mentioned paper, a Digital Triad is proposed as an advancement beyond DTs, where design knowledge and application-specific intelligence is encapsulated in a very similar way to AAs. Surveying all of this "submerged" literature is a complex task given that terminology likely do not match. However, we hope that our contribution can promote cross-fertilisation with these related research efforts, in a joint effort to avoid reinventing the wheel—our main motivation for sticking to AAs instead of introducing brand new concepts into DTs.

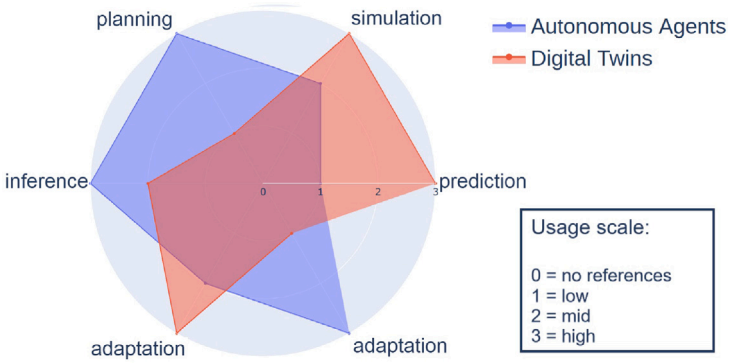


Fig. 1. "Coverage" of intelligent functionalities by AAs and DTs, as stemming from the related literature. As advocated in this paper, AAs and DTs appear as complementary abstractions.

2.5. Intelligent functionalities for AAs and DTs

The literature on distributing intelligence in IoT discussed in previous sections reports on several intelligent functionalities for which AA and DTs are being actively used by system designers. We summarise and categorise them here regardless of the specific task they accomplish (e.g. time series forecasting vs. fault prediction) and of the specific technique adopted (e.g. regression, SVM, etc.), but focussing on the *kind* of intelligent function they deliver. This categorisation is useful first of all in defining what we can consider, for the scope of this paper, "intelligence" in a practical, bottom-up way, stemming from related works in the area without getting trapped in philosophical arguments. Then, it is also useful to establish the *coverage* of such functionalities by AAs and DTs, as shown in Fig. 1, which already suggests that they can be used synergistically to deliver the full spectrum of such intelligence.

- *Prediction*, there including time series forecasting, recommendations, namely any form of reasoning meant to "guess" new information based on past and current knowledge. In the IoT, common prediction targets are machinery failures, stock levels, resource use, and system states.
- *Simulation*, encompassing any form of reasoning by hypothesising different states, in the past, present, or future. "What-if"-like analysis falls into this category. In the IoT, it is common to simulate the future states of individual things, for example, to improve the design of a product or a production pipeline. However, complete systems can also be simulated with an added degree of complexity.
- *Planning*, that includes any form of synthesis, and *practical reasoning* [101], which is meant to figure out how to achieve some practical effect (on the target system) by properly sequencing available actions. Planning to achieve a given system configuration is common in the IoT, as well as planning to reconfigure after some disruptions.
- *Inference*, within which we include both *epistemic reasoning* [102], that is, synthesising novel information from known data, or data-driven inference such as pattern recognition and classification. This is perhaps the most common form of intelligence used in IoT, as even simple monitoring and control applications usually infer situations by aggregating different data coming from distributed sources of information. In addition, fault detection and diagnosis can be gathered in this category.
- *Adaptation* instead gathers all the intelligent functionalities aimed at enabling the system to adapt to unforeseen situations that had not been explicitly managed at design time. As in the previous category, this one is quite broad and includes several heterogeneous approaches, ranging from engineered adaptation (e.g. the MAPE-K loop [103]) to learning-based methods (e.g. evolutionary approaches [104]).
- Finally, we highlight the role of Learning, which despite not usually being the primary goal for a functionality, is a valuable tool to implement some of the ones highlighted above and still poses important requirements on the architecture of a system that wishes to support any form of learning process in one of its components. That includes statistical machine learning, reinforcement learning, and causal structure learning [37,105–107]. In this category, we group the functionalities that aim to make the system, or one of its components, learn to do something. In IoT, the most common form of learning employs statistical machine learning to create prediction models, for example.

The next Section refers to these categories of intelligent functionalities to illustrate why and how to use the design principles proposed therein.

3. Principles and architectural perspective

Given the current interpretation and use of the AA and DT abstractions in the literature, it is apparent that both AAs and DTs can be used to encapsulate some form of intelligence. Thus, an open engineering question involving both research communities is: are AAs and DTs different ways to achieve essentially the same thing? Are they (possibly partially) overlapping? Or are they complementary?

In this section, we argue for the latter. We object that existing approaches often blur the boundaries between the responsibilities of AAs and DTs creating confusion by trying to fit everything within the same abstraction (be it an AA, a DT, or any other unprincipled *ad-hoc* combination of the two abstractions). The direct cause for this is that the AA and DT abstractions are not associated with clear roles within the architecture of an IoT system, hence they do not help produce *modular* and *reusable* designs that can then be implemented across application domains. This generates several competing architectural models all trying to address similar problems and thus overlapping with existing solutions, leading to fragmentation and confusion for researchers and practitioners seeking the "best" design for their specific problem at hand.

In an attempt to remedy this, we first identify a set of principles to assist system designers in the analysis of the functional requirements of their intelligent IoT system at hand (Section 3.1), then frame AAs and DTs as the two abstractions perfectly adhering to such principles, complementarily (Section 3.2). Finally, we propose AAs and DTs as the basic building blocks that designers can use and compose to create their application-specific software architectures (Section 3.3).

3.1. Design principles

We take our move from the quite abstract software engineering criterion of *separation of concerns* given in [8], expanding and refining it along three dimensions to make it more practically useful: *specificity*, *scope*, and *time*. These are meant to be used to analyse the requirements posed by the intelligent functionalities summarised in Section 2.5. We anticipate that it is likely that neither of these dimensions *alone* is sufficient to determine *univocally* which abstraction is best suited for a given functionality. Instead, we argue that they all need to be taken into consideration together when approaching the design of an intelligent IoT system.

The reason to ground our principles in a *software engineering* standpoint, that is, from the perspective of the software designer of an IoT system or application, is that IoT systems are increasingly no longer "mostly hardware" ones, with interconnected devices that simply exchange data. Instead, they are increasingly becoming complex cyberphysical systems where software plays a prominent role, especially regarding endowing intelligence into devices (hence into the system). There, thus, proper engineering of the software layer of the overall IoT deployment is crucial to guarantee both functional and non-functional properties. Not by chance, the publications discussed in Section 2 deal mostly with *software* engineering, architectures, and design.

3.1.1. Specificity

The first principle we propose is what we call *specificity*, whose spectrum is depicted in Fig. 2, and is defined informally as

> "how much a given (intelligent) functionality is *specifically* serving a given application goal (at one end of a spectrum), rather than being *generally* exploitable by multiple goals and potentially across application scenarios (at the other end)."

With this, we want to help IoT systems and application designers to distinguish between *(i)* those features that are strictly tied to the achievement of a specific *application goal*, and *(ii)* those that may instead be *generically* useful to pursue multiple goals and build other intelligent functionalities and/or applications on top. This is especially important for designing highly modular systems.

In IoT systems, for instance, a widespread general-purpose functionality concerns the digitalisation of Physical Assets (PAs) to enable intelligent functionalities such as prediction of that assets' future states, and simulation of alternate states. Functionalities like these can be considered to be not tied to the application goals, but to the general services offered by the IoT platform, in an open systems, "as-a-service" perspective [18]. In contrast, functionalities such as fault diagnosis or inference of specific information may be relevant only in the context of the application domain or goal. Another example would be the implementation of adaptive control policies and decision-making strategies: these are likely to be specialised for the system at hand, as they encode the requirements and goals set by stakeholders.

However, each specific instance of these categories of intelligent functionalities devised out in Section 2.5 must be carefully examined by the system designer in light of this (and the other) principle(s), as there can be no "rule of thumb" generally applicable to every application domain and goal. Also, generally speaking, there may be edge cases that may feel "in violation" of our proposed principles but are actually not. For instance, it may be the case that an application goal coincides with the digitalisation of an asset for prediction purposes. This case should not be interpreted as "wrong" because it defines a prediction function as application-specific. On the contrary, the specificity principle, in this specific case, cannot guide designers' choice *alone*. This is one reason to propose more, complementary principles, that better capture the different nuances of complex systems and applications—not by chance the edge case described is quite simple.

Fig. 2. The spectrum of *specificity*: from intelligent functionalities specific to an application goal to general-purpose services.

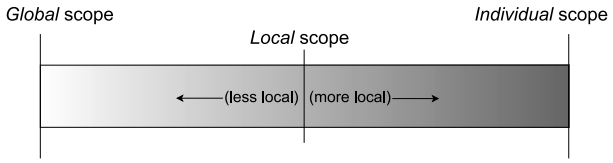


Fig. 3. The spectrum of *scoping*: from intelligent functionalities requiring knowledge of the whole system, to those confined to individual assets.

3.1.2. Scoping

The second principle we propose, we call *scoping*, is defined as follows:

> "to what extent a given (intelligent) functionality requires data from the whole system (at one end of a spectrum) or a single individual entity of interest (at the other end) to deliver its results"

With this, we want to help IoT systems and applications designers distinguish between *(i)* the features that require data, inputs, and interactions with multiple entities of interest in the system at hand (up to *global* information, synthesised at the system level), and *(ii)* the features that instead require only considering a single asset (the extreme case of *local* information). Fig. 3 depicts this spectrum.

In any IoT system, due to its inherent connection with the physical world, there are software components whose main function is coupled with PAs, which set boundaries to the *scope* within which intelligent functionalities operate. For example, by collecting local information only (e.g. prediction of the future states of specific machinery, not others). Conversely, many other functionalities may require a broader scope that goes well beyond such local boundaries. If we take as an example the domain of a smart home, we can have software components that mirror, and grant access to, the individual devices (e.g. for remote monitoring and control), but also adaptive control routines spanning multiple devices or even multiple rooms to orchestrate a coordinated action (e.g. preparing for the comeback of an inhabitant by activating the A/C, unlocking the doors, raising the curtains, etc.). Accordingly, with the scoping principle, we suggest that system designers consider where the knowledge and data required to accomplish the intelligent function come from, as well as which entities will be subject to its effects.

It should be noted that the *individual* and *global* scopes are the two extremes of the spectrum depicted in Fig. 3. Functionalities may require data from multiple entities, but such entities may be confined in a limited space, for instance geographically, or network-wise. Or, it could be the case that data is required from different and distant entities that are anyway somehow easy to access together—think of overlay networks. In this case, the proposed principle is still useful to guide design choices, as it forces designers to clarify and sort out why and how they deem the scope to be worth defining global or local (or even individual).

3.1.3. Timing

The third (and last, for the time being) principle we propose is what we call *timing*, which we define informally as

> "how much a given functionality relies on any one notion of *time* (e.g. physical or logical) to be either explicitly (i.e. *time-aware*) or implicitly (i.e. *time-situated*) available"

By "time-aware", we mean that the functionality requires access to time-related information, either to make it directly observable to consumers of such functionality or to work properly. For instance, time series forecasting (a form of prediction) needs an explicit representation of time to be available, hence it is time-aware. By "time-situated", instead, we mean that the functionality has some dependency on any one notion of time, but such dependency need not be explicitly captured and managed. For instance, real-time monitoring and control of an asset surely depend on time, but such a dependency need not be modelled to deliver the functionality. Simply, its existence suffices for the functionality to work properly. There are also some other cases where time does not matter from the designer's standpoint, as time is not an issue when implementing the functionality—e.g., sending commands to an actuator device. This whole spectrum is depicted in Fig. 4.

Of course, placing intelligent functionalities within this spectrum cannot be a precise operation, as with the other principles. But some general considerations about the categories of intelligent functionalities devised in Section 2.5 can and should be made to aid system designers. For example, prediction, simulation, and diagnosis are likely to require time awareness. How far into the

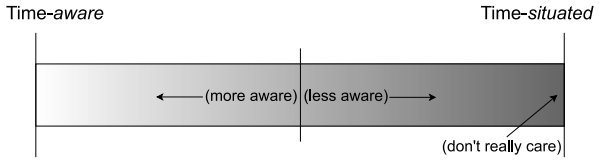


Fig. 4. The spectrum of *timing*: from intelligent functionalities requiring explicit modelling of physical or logical time, to those that do not care.

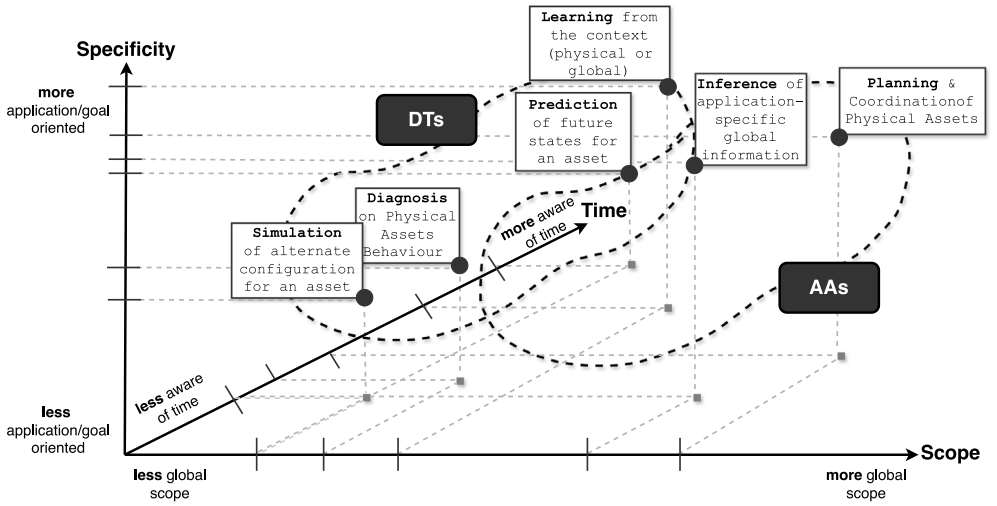


Fig. 5. Exemplary intelligent functionalities placed within the spectrum of our design principles. AAs and DTs are clustering those functionalities that they better match with, according to such principles.

future should an event or state of interest be predicted? How long into the future should a simulation extend? Conversely, when did a chain of faults lead to the present faulty situation? Other functionalities, such as inference, learning, and adaptation, may be less demanding and simply be time-situated. Of course adaptation actions are based on some events that happened, or on some predicted future state, and need to be carried out in the future, but possibly they do not require an explicit representation of time to work properly as simply their sequential ordering is sufficient to eventually react accordingly.

3.2. AAs and DTs as first-class abstractions

In the previous subsection, we propose our design principles guided by the intelligent functionalities that are described in the related literature discussed in Section 2. Now we match such principles against AAs and DTs, to conceptually assess which abstraction is best suited for which end of the spectrum described above for a given principle. Fig. 5 shows the exemplary functionalities and how they are better encapsulated by AAs or DTs according to our principles.

Let us start by considering how AAs and DTs have been used in the *specificity* design principle. AAs, being by definition *goal-driven* entities, are usually exploited to encapsulate intelligent functionalities that are strictly related to the problem to solve, hence to the domain and objectives of the application [108]. DTs, instead, are usually exploited with the scope of digitalising assets to provide data-oriented services to consumer applications, *servitisation* being one of their core characteristics [109]. The principle of specificity, thus, would encourage designers to encapsulate intelligent functionalities closely matching application goals in AAs, and, complementarily, to model general-purpose services as DTs instead.

Consider now the *scoping* design principle. AAs are not tied to any particular source of information in achieving their goals. Although part of the definition of AA includes a relationship with the notion of an environment in which AAs are *situated* [29], AAs are not limited in any particular way to any given "object" in such an environment. In other words, the application environment (physical or digital) in its entirety is a source of information that AAs can perceive and (possibly) affect—their *global* scope. DTs, instead, are specifically defined as being *coupled* with the individual asset in the physical world that they are digitally representing (their physical twin). It is thus natural to give DTs a *local* scope, limited to the information available from this physical asset. Note that we deliberately choose the term "scoping" instead of, for instance, "locality", to avoid the misunderstanding that such design principle is inherently tied with some notion of space, such as in a geographical sense or limited by a physical size boundary under which something can be considered local or not.

Finally, let us position AAs and DTs with respect to the *timing* design principle. AAs do not include any explicit notion of time in their definition. The situatedness feature already mentioned includes the temporal dimension but does not prescribe agents to capture and model time-related aspects. Hence AAs are *time-situated* entities, naturally. DTs, complementarily, are specifically requested to keep in synch their related PA to provide an updated digital representation in a timely manner. This implies explicitly capturing the time at which events in the PA happened, and modelling the temporal evolution of the PAs. DTs are thus naturally *time-aware*.

Summarising the above considerations, AAs and DTs can be described as playing the following roles in the design of distributed intelligent functionalities in IoT systems and applications:

- **AAs** are the components of the IoT system that encapsulate the system/application goals, hence are aware of such specific goals and strive to achieve them by autonomously deciding whose other entity to interact in the whole application domain. AAs are also not particularly tied to any specific notion of time.

- **DTs**, instead, model and encapsulate properties, behaviours, and functions of specific portions of the IoT system or application domain, therefore are inherently bound to specific assets, to provide general-purpose services to other components or directly to the application. Due to such modelling, DTs must at least account for the specific notion of time relevant to the modelled asset.

We emphasise that these definitions are not meant to set crisp boundaries amongst AAs and DTs in any possible use case and scenario. Consequently, there is a degree of flexibility in how these definitions can be interpreted to distribute responsibilities across different components. However, having well-defined criteria that dictate how to exploit AAs and DTs as complementary abstractions helps to identify responsibilities in the design of an IoT system willing to fully leverage distributed intelligence.

To better illustrate this flexibility, the next subsection describes some architectural solutions that can arise from these design principles to deliver different intelligent functionalities in the IoT. Furthermore, Section 4 discusses a practical case study, applying the principles to analyse the system requirements.

3.3. From "macro" to "micro" architectures

The proposed design principles and their match against AAs and DTs help decide whether an AA or a DT is best suited for a given intelligent functionality. However, they tell little about how to combine AAs and DTs *synergistically* when intelligent functionality requires it. For instance, because it is positioned within the spectrum of each principle in a way that does not perfectly match all the features of either an AA or a DT, but some of both. Accordingly, this section discusses the several "micro-architectures" for AAs and DTs integration that may arise during the application of our principles—while the next section provides a practical example for a selection of them. We use the term *micro-architecture* (μ-arch, for short) to emphasise an important distinction our architectural perspective has concerning the related literature, discussed below.

The goal of clarifying integration architectures is also witnessed in recent literature, although from different perspectives. Ref. [8] enumerates different alternative architectures to integrate AAs and DTs, depending on the goal of the integration itself. In [10] the goal is to achieve a technical integration between specific technologies to increase the level of abstraction toward the set of concepts most familiar to the developers of AA. Another effort, the *Web of Digital Twins* (WoDT) vision [18], sees DTs as entities interlinked in a *web of semantic, dynamic relationships*, working as the digital substrate that enables structuring a dynamic application domain. According to the WoDT vision, a layer of networked DTs works as the interface between applications (either agent-oriented or not) and the physical environment they must cope with, thus decoupling the two layers and possibly providing augmented and cross-domain functionalities.

Common to these and other architectural proposals in the literature [55,110–113] is the *top-down*, "macro-architecture" perspective adopted: what is proposed is a reference architecture for a whole system, a blueprint to adopt in any given IoT scenario (and beyond). For instance, in the field of enterprise systems architecture, the rise of IoT and the relevance of CPSs in general as the "backbone" of many intelligent functionalities within the company (that require asset monitoring and control first and foremost), nurtured research in new designs and methodologies supporting IoT-related goals. There, the general trend is to adopt a *System of Systems* (SoS) perspective over the whole enterprise, decomposing the requirements along the same application-agnostic dimensions. The FCBPSS architecture is an example [114], where the many systems supporting operations of an enterprise are seen in terms of Functions, Context, Behaviour, Principles (guiding the design), Structure (components realising the function and their relations), and State. These very same concepts are applied recursively for the whole enterprise integration system—indeed, a SoS. In contrast, here we propose a bottom-up approach, where the architecture of the overall system is the result of the composition of multiple μ-arch, depicted in Fig. 6 by making use of the following components: Physical Assets ("PA" squares), which represent the entities in the real world (e.g. devices, objects, people, processes, organisations) that the IoT system needs to model; DTs ("DT" circles); AAs ("A" circles); and the intelligent functionalities (the 3D boxes), which may be entire applications or simply one of the many functions needed for the application at hand. The possible μ-archs are:

- μ-arch **(A)** is the most common in the related literature already mentioned [8,10,18], and it is widely used as a reference architecture for the integration of AAs and DTs [8]. There, the intelligent functionality to realise neither perfectly matches an AA nor a DT. For instance, it could have high specificity, require time-awareness, and a scope locally extended to a set of related PAs. Hence, PAs are digitalised by DTs, that offer services to AAs encapsulating the intelligent functionalities directly serving the applications' goals. This is the typical μ-arch that arises when a given intelligent functionality can be decomposed into an application-specific part and a PA-specific part [8]. The former can request to complement the PA-related information with external data, and directly serve the application goals—hence is better encapsulated by an AA. The latter may have temporal constraints on the PA and can be reused across domains—hence is better modelled by a DT.
- μ-arch **(B)** encompasses instead an edge case in which the intelligent functionality can be directly mapped onto a DT: it has low specificity, its scope is limited to a given PA (or cohesive set thereof), and time-awareness is required. In this case, the AA abstraction is unnecessary as the nature of DTs makes them suitable to deliver the functionality alone.
- μ-arch **(C)** is quite common in AAs literature, and follows the "agentification" paradigm [3,115]: wrapping any relevant object as an agent, to model the domain as a Multi-Agent System (MAS)—physical devices and objects included. Again this is an edge case that needs to be treated with care as it can lead to undesired coupling. We argue that this is legit when the goal of the agent is specific to the asset and the scope of its decision-making is strictly localised. Such a goal should not be to serve other application components with a digital representation of the asset of course – that would be the true nature of a DT instead –

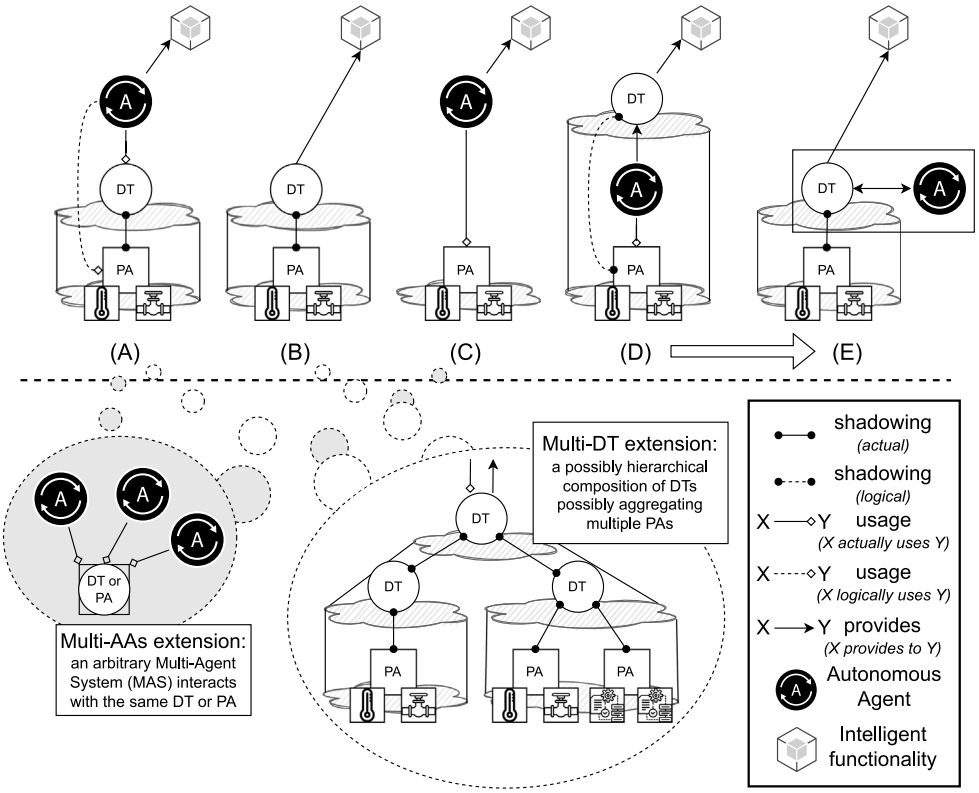


Fig. 6. Architectural perspective for the synergistic combination of AAs and DTs in distributing intelligence in IoT systems. Each letter denotes a specific *micro-architecture* that may arise from the application of our principles. Such architectures can be combined to give shape to the overall system architecture, in a bottom-up way.

but can, for instance, represent a closed feedback loop on the asset itself. Similarly, whenever the scope of the functionality becomes "larger" such as in the case of coordination with other agents, we argue that μ-arch **(A)** is more effective in decoupling the interaction with other agents from the control on the asset through the corresponding DT. We observe then, that with the rising complexity of a system this approach usually often degenerates in either **(A)** or **(E)**.

- μ-arch **(D)** is one we actually argue against. Such an architecture is common, for instance, in the literature about agent-based DTs [116,117]. Also, MAS-based DTs [17,118] fall in this case, if more AAs are used. The issue here is that the mirroring, or shadowing, of the PA by the DT seemingly has to go through an agent. This is likely to cause issues, such as delays, in the process itself [119]. Moreover, as stated before, it should not be the responsibility of an AA to digitalise a PA. We argue that a better replacement for a combined solution considering AAs and DTs is depicted in μ-arch **(E)**, described below.
- μ-arch **(E)** simply expresses in a slightly, but impactful, different way the real intent behind μ-arch **(D)**: *augmenting*, enhancing the capabilities of a DT. Thus, we argue that a better depiction of this need is the one where the DT still is responsible for digitalising the PA, but interacts with an agent in the process to implement or expose augmented capabilities to the consumer components. Differently from **(A)** the agent in this pattern *disappears* within the DT and the other components of the system cannot interact with it directly. This is in line with the definition of a DT that externally is seen as the component that mirrors a PA while adding specific functionalities to it, but internally keeps separation of concerns between the digitalisation process and the other functionalities that are better captured with agents (e.g., decision-making, planning, simulation).

Most of these μ-archs can be trivially extended to the multi-AAs or the multi-DTs case, where not one but multiple AAs or DTs are used. These cases are exemplified at the bottom of Fig. 6. Essentially, every time a single AA or DT appears in a μ-arch, that AA or DT can be extended to be a multi-AAs/multi-DTs subsystem. An example already used is that of the DT of a complex structured PA (a room, a whole building), which can be obtained by suitably composing multiple DTs in a hierarchy [120].

Finally, we emphasise that these μ-archs can be arbitrarily composed in "recursive structures" to give shape to complex IoT systems [118]. Let us assume, for instance, that a complex DT is set up to digitalise a whole hospital ward. The DT is the composition of multiple DTs, each shadowing a specific room, each in turn shadowing every single equipment. Some of these DTs can provide general purpose services and can be assisted by AAs according to μ-arch **(E)** in doing this. Then, these DTs may be exploited by a multitude of AAs as per μ-arch **(A)** extended to the multi-agent case, each exploiting their general purpose services while providing their application-specific functions. By iterating this recursive composition of μ-archs, the overall IoT system architecture

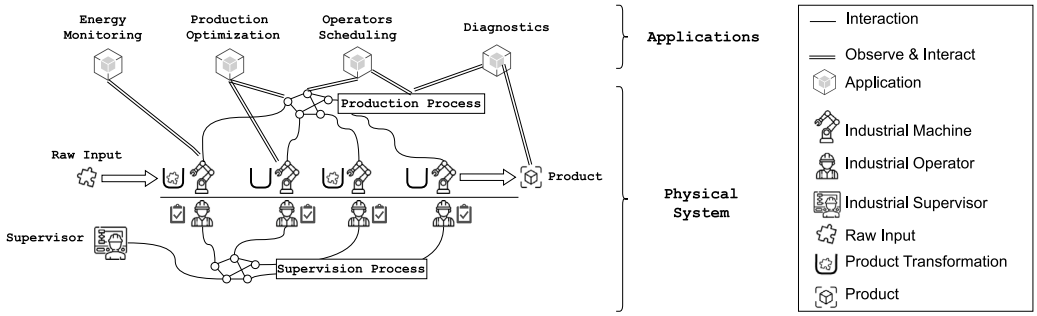


Fig. 7. A reference Smart Manufacturing scenario with multiple physical entities and a set of applications interested in implementing intelligent functionalities on top of the physical deployment.

emerges in a bottom-up way as dictated by the nature of the intelligent functionalities themselves, not as imposed by a reference architecture—which may be unable to capture the specifics of each different domain and application.

The next section illustrates how both our proposed design principles and the resulting μ-archs can be applied in a practical case study in the domain of smart manufacturing.

4. Exemplary application of the design principles

In this section, we exemplify the application of our proposed design principles and the resulting micro- and macro-architectures, in the case of a smart manufacturing scenario. For the sake of exemplifying the design process we can imagine having a goal of implementing IoT systems capable of dynamically optimising production about the overall energy consumption of a manufacturing plant. This goal can be achieved by designing a distributed intelligence system based on a combination of AAs and DTs that implement a set of high-level functionalities (which we recall from Section 2.5) that include monitoring, prediction, and planning for dynamic task allocation, and adaptation to external factors.

We unfold the design process by first analysing the functionalities from a domain-related perspective, decomposing the main goal into more fine-grained functionalities that can be directly implemented. Then we apply the principles described in Section 3, to identify the main characteristics of such functionalities and finally decide on the combination of AAs and DTs accordingly. In doing so, we provide a guiding interpretation of those principles when confronted with real-world problems and requirements. We believe that this further validates how having such common abstractions in the form of AAs and DTs can contribute to the design of complex distributed intelligence systems.

4.1. A smart manufacturing energy optimisation scenario

A typical production system distributes a diverse array of *machinery*, *operators*, and *processes* throughout the shop-floor environment (as schematically represented in Fig. 7). These individual components are often organised into *production nodes*, which serve as logical groupings of related machines. Several nodes make up the overall production plant.

Industrial energy monitoring [121,122] and optimisation [123] is a challenging task, especially in a distributed environment. There, such tasks might involve several processes, to selectively turn on different nodes when needed, depending on the external information about the energy cost, while keeping the production speed at an acceptable rate. Given the hierarchical organisation of the plant and the cyber–physical distribution of the system, this problem calls for a DI approach.

As summarised in Fig. 8, we decompose the main target of production optimisation into three objectives that could be strategic in the target scenario and that represent a useful reference to analyse and apply the design principles presented and proposed in this article.

We identify three main functionalities that concur to dynamically adjust production based on energy costs, to keep the overall consumption under a threshold. The IoT system must be able to collect information about the current energy consumption and needs of each production node through some form of *energy monitoring*. Furthermore, to be able to dynamically adjust the loads in the production nodes, some form of *intelligent coordination* is required to continuously optimise the utilisation of resources. Finally, the overall *production optimisation* techniques are required to make high-level decisions that incorporate data from both individual production nodes and external sources. These functionalities are an exemplification of decomposition that can be performed in the design process of an IoT system. In the following, we analyse each functionality in more detail and guide this analysis toward the definition of components that can implement the IoT system employing either DTs or AAs or a combination of the two.

Energy monitoring. To make dynamic adjustments to the production process based on the energy consumption of the whole plant it is essential to be able to monitor it in real-time through smart metres embedded in the machines. Furthermore, it is essential to collect data from each machine or production node to have fine-grained control on what parts of the plant are consuming the most energy. Monitoring can also include some forms of inference to determine whether the observed consumption is considered regular or is an anomaly, and potentially take corrective actions in such cases to improve safety. Finally, prediction or simulation can be used to forecast what the energy consumption is expected to be, given some production tasks, to improve decision-making.

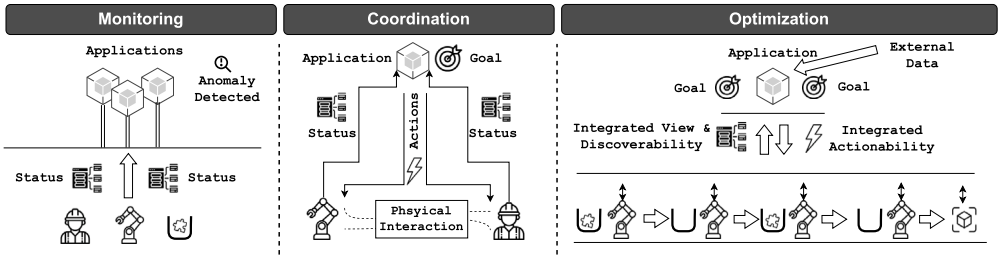


Fig. 8. Three main objectives that concur to the energy optimisation of a Smart Manufacturing plant: Energy Monitoring at the machine level, Intelligent Coordination between robots and operators, and overall Production Optimization based on external factors (e.g. energy cost).

Intelligent coordination. This objective involves the use of data collected from the IoT system to introduce coordination capabilities to improve both performance and safety at different levels of deployment (e.g., machine-to-machine or machine-to-operator interactions). For example, machine learning models can be trained to detect patterns and, by continuously monitoring and analysing data streams from machines and operators, the system can automatically understand the context of the currently performed activities. This is essential to understand the best planning policy to schedule the next tasks as soon as machines or operators are free to perform them. For example, if a portion of the plant is to be shut down for energy saving, the tasks that were queued on those machines should be redistributed across the other active production nodes of the plant.

Production optimisation. Digital applications can combine real-time data from the physical world and external sources (e.g., market price and production demand) to dynamically optimise manufacturing processes together with all machines, operators, raw materials, and products. This requires decision-making driven by inference of constraints, as well as means to monitor the current situation to adapt according to the real-time state of the system. For this scenario, we assume that the market price of energy is provided by an external forecasting system. Based on that and according to target business rules, the digital layer can optimise production schedules, resource allocation, and task assignment to maximise efficiency and reduce costs.

In the subsequent section, we explore how AAs and DTs can be synergistically exploited to construct a cyber–physical intelligent abstraction layer following the design principles presented in Section 3.1.

4.2. System design with AAs and DTs

Table 3 reports the mapping between the intelligence functionalities envisioned, the application of the design principles, and the associated micro-architectural design. We decompose each overall objective into specific functionalities and associate them with the main *kinds* of intelligent functionalities identified in Section 2.5. The analysis leads to the adoption of an AA, a DT, or a combination thereof, for functionality, and a different μ-arch for each objective (as shown in Fig. 9), choosing from those presented in Section 3.3.

In the following, we discuss the reasoning behind each choice, providing an illustrative interpretation of the principles for the different functionalities in the scenario presented above.

Energy monitoring. For this objective, we identify the basic functionality of data collection about the energy consumption of a given machine. It has low specificity as the very same data required for the monitoring could easily and likely be used for other, future tasks (e.g., prediction of consumption). Of course, as the data belong (is both generated and consumed by) to a specific PA, the functionality is *local* concerning the Scoping principle (actually, individual), and we consider an explicit time representation needed for this task to generate a time series.

We also expect to have an anomaly detection functionality that can analyse time series data and detect unexpected patterns. The knowledge required for such a task is still local to the machine, and the results can be useful to general services as they could notify interested parties of a malfunction.

Finally, we envision an on-the-fly corrective reaction mechanism that is introduced with the specific purpose of shutting down the machine if there is a problem. This is a specific behaviour introduced for our main monitoring application goal (hence, with application goal specificity), which still uses only the local data and can act on an event-driven trigger that does not necessarily need an explicit time representation.

From this analysis, we can reliably identify the need for a machine DT, capable of collecting data and detecting anomalies, and of an AA implementing the emergency shutdown policy. This leads to μ-arch **(E)** described in Fig. 6 as the possible reference implementation for this functionality.

Intelligent coordination. The data necessary for this functionality comes from an activity monitoring task, which has *local scope* on the PA being mirrored (e.g. an operator) to infer the activity currently executed and possibly predict the expected duration (thus, we an explicit representation of time needed, here). Then, each machine or operator should also expose a digitalised and personalised functionality to receive the next tasks for delegation. Such function would be assigned from dynamic planning performed by another component of the system, after receiving the aggregated global knowledge (hence global scope) from each entity involved in the coordination scenario.

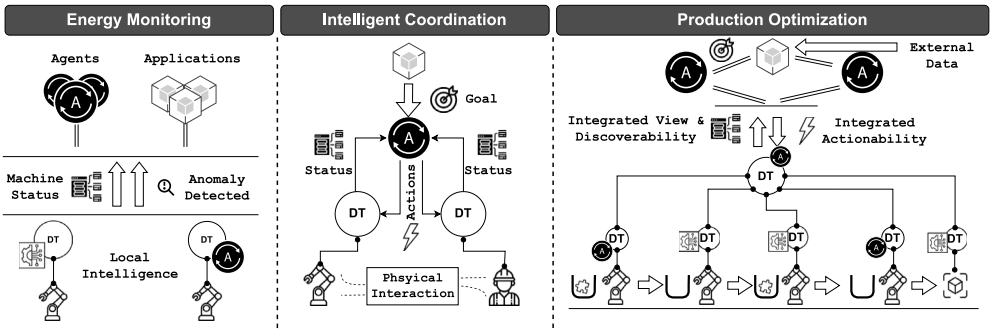


Fig. 9. Three illustrative instances showcasing the synergistic usage of AAs and DTs as guided by the principles and μ-archs, within the Smart Manufacturing context.

Accordingly, we may expect that each PA would have a DT that mirrors them and encapsulates the activity monitoring and task delegation functionalities, tailoring them to the mirrored entity. Planning, instead could be performed by an AA keeping track of the global context (e.g. of a whole production node) and applying a specific coordination policy to redirect tasks to different machines (hence, a functionality with goal specificity and global scope, justifying preference for an AA). This leads to the adoption of μ-arch **(A)**, with a single agent using multiple DTs as its data sources and delegating tasks to each.

Production optimisation. Optimisation at the plant level will be guided by a decision-making functionality based on two sources of data: external data about the market price of electricity for a specific time and date and data collected locally from the different production nodes. Depending on the cost of electricity, for instance, the policy may decide to shut down the nodes that are not actively involved in the ongoing process, delegating tasks to different nodes.

Accordingly, μ-arch **(A)** is the best suited in this scenario, involving a DT that mirrors the production process as the main data source, and an AA enacting the policy and acting on the production nodes. It is worth noting that, having already introduced the DTs of the different production nodes, we can also consider linking the DT of the production process to the DTs of the nodes, in a composition pattern that would facilitate the modelling of the production process DT as represented in Fig. 9.

4.3. Discussion of resulting architecture

Fig. 10 schematically illustrates the integrated architecture, using both AAs and DTs, resulting from the application of our proposed design principles and the consequential composition of our μ-archs.

The physical world, composed of multiple heterogeneous entities such as machines, operators, raw inputs, and products, can be effectively digitalised through the use of DTs. These DTs are responsible for representing the associated PAs in cyberspace through interoperable and homogeneous digital replicas, operating locally to the devices and associated just to their context without the need for a global scope. This approach can also be applied to industrial processes, such as product transformation from raw inputs to final products, the interaction and collaboration between machines and operators, etc.

DTs excel in abstracting and decoupling PAs and processes into modular digital representations, simplifying interaction and integration. They support standardised communication protocols, enhancing interoperability across various platforms and systems,

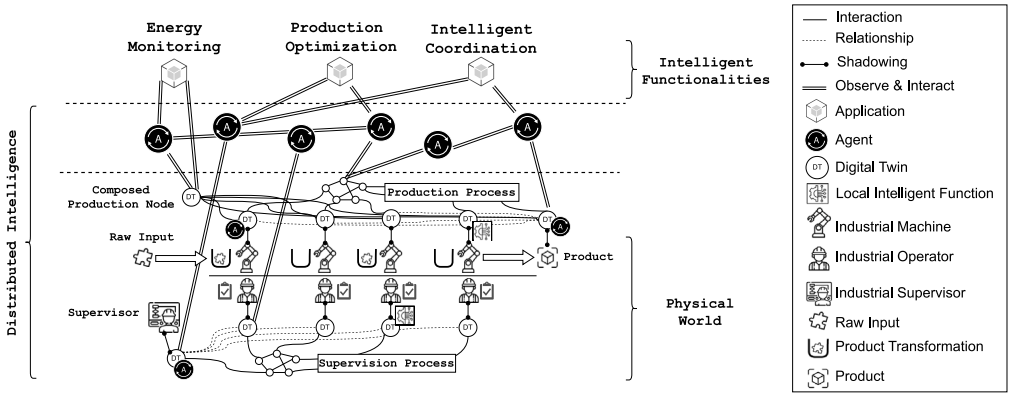


Fig. 10. A layered architecture with the combination of DTs and Agents for a Smart Manufacturing reference scenario.

that is critical for integrated manufacturing operations. DTs can be composed of higher-level aggregates, providing a holistic view of the system, which is crucial for understanding interdependencies and optimising system-wide processes. Moreover, DTs can embed *localised intelligence*, enabling real-time monitoring, anomaly detection, and predictive maintenance, while reducing latency and improving responsiveness.

On the other hand, AAs are designed to operate autonomously, making decisions based on predefined rules or learned behaviours, which allows them to adapt to changing conditions and respond effectively to real-time events. They excel in tasks requiring dynamic coordination and optimisation, such as reallocating resources and adjusting schedules based on real-time data and system conditions. Additionally, AAs can be easily scaled and deployed across different parts of the manufacturing system, providing a flexible solution that can grow with the system's needs. In the envisioned and reference smart manufacturing example, AAs can be distributed across the entire deployment spectrum. They support both local inference and decision-making within a DT, allowing it to achieve its internal target goals, and broader spectrum operations enabling dynamic coordination throughout the production line. This includes adapting planning and task allocation in response to variations in the production line context or external market demand.

AAs and DTs also benefit significantly from each other, in a perfect synergy. AAs benefit of DTs decoupling of complexity when interacting with the physical world and the exploitation of a uniform and interoperable digital representation of it. Conversely, AAs bring intelligence to the system, operating either internally as components of the DT's model (both for single and composed twins), or on a broader scope utilising the different DTs to collect information and act to improve overall performance and behaviour. An additional benefit of such an integrated approach that combines both DTs and AAs is that the components can evolve separately. Policies can change at the AA level (even online, via learning) without having to modify the DTs. Similarly, new and improved DT models could be produced as data is collected about the system, improving support given to the AA decision-making, without changing the AA at all.

As a final remark, emphasising modularity, it is worth noting that in this example the decision-making is all performed by AAs. This means that in case of need (e.g. some critical failure or an expected contingency), to switch the system to a *"manual mode"* it is sufficient to momentarily stop/disconnect the AAs. Then, they can be temporarily replaced by human operators to amend the system as needed, while still benefiting from the real-time data collected by DTs that can continue to operate.

All of these desirable non-functional properties do not come without costs: the next Section discuss the challenges that researchers and practitioners need to deal with to effectively adopt AAs and DTs according to our proposed principles and μ-archs.

5. The road ahead

To fully harness both AAs and DTs in shaping DI within IoT systems according to our proposed principles and μ-archs, a few open challenges still need to be addressed.

5.1. Interoperability & standardisation

One of the primary challenges is ensuring the interoperability of DT technologies across various systems and platforms. This is especially relevant in our view, where DTs are seen as mostly in charge of realising general-purpose services, reusable across application domains, in an open systems perspective. As DT technology evolves, there is a risk of fragmentation where different industries and organisations develop their own DT solutions *in isolation*, without an ecosystemic and interoperable vision.

To address these challenges, academia, industry, and standardisation bodies should collaborate to develop standardised and interoperable DT specifications and frameworks [65]. This effort should promote interoperability across various solutions and domains by establishing common data formats, uniform DT descriptions, interoperable communication protocols, and interaction models. In addition, creating unified ontologies and vocabularies will ensure consistent data representation and interpretation across various DT implementations.

AAs, instead, can rely on the FIPA set of standards [124] for interoperability, also at the semantic level. However, the peculiarity of IoT scenarios is likely to require a revision of such standards.

5.2. Modelling of intelligent functionalities

With the envisioned distribution of intelligence across DTs and AAs, the representation and description of intelligent capabilities, and consequently their interoperability, represents a critical challenge. In this paper, we deliberately stayed away from discussing the specific techniques and mechanisms available to date to implement intelligent functions. Statistical machine learning, reinforcement learning, causal reasoning, cognitive agent architectures, etc. are all different means to implement the kind of intelligent functionalities that we summarised in Section 2.5. However, each comes with its preferred representation of the input and output information. To seamlessly encapsulate and compose these implementations in AAs and DTs that may successfully cooperate, requires dealing with aspects such as the adoption of various formats, representations, and ontologies, together with issues related to discoverability, interaction patterns, and information retrieval.

Doing this should maintain a level of compatibility with existing efforts in the IoT domain such as the Web of Things,[1] oneM2M,[2] or ontologies such as SAREF.[3] The goal is improving the ability to describe complex entities such as AAs and DTs, with their

[1] Web of Things: https://www.w3.org/WoT/

[2] oneM2M: https://www.onem2~m.org/

[3] ETSI SAREF: https://saref.etsi.org/

properties and intelligent features, to promote the interoperability of these components with open and dynamic IoT systems. A domain-independent, structured, and interoperable description of intelligent capabilities, applicable across different entities and distributed deployments, will ultimately enable the definition of an *intelligence continuum* decoupled from specific implementations and domain-driven platforms. This continuum aims to align intelligent functionalities under a coherent, interoperable, and distributed overlay that can be used as-a-service by other components, minimising the number of prerequisites and manual configurations.

5.3. A meta-model for an intelligence continuum

The definition of a representation for the different categories of intelligent functionalities we summarised in Section 2.5 is half the story: the other half would be defining a meta-model for the seamless interaction of such different representations. That is, a meta-model for the *intelligence continuum* enabling to seamlessly "jump" from a given representation to another. For instance, to feed a BDI agent [125] with a standard structured representation of the predictions of statistical machine learning algorithms in terms of beliefs or plans.

A similar meta-model has been defined in terms of events in the Web of Digital Twins vision [18]. There, the meta-model described how to represent DTs' functions and operations from an open systems perspective. Here, a similar meta-model should serve as a practical aid in designing the intelligence continuum across the many paradigms available nowadays, such as agent-oriented programming, statistical machine learning approaches, and the rapidly rising large language models—to mention a few. The kind of event structures we foresee bear resemblance to Dynamic Bayesian Networks, Probability Trees, and Causal Directed Acyclic Graphs [126], and these domains already offer methods to navigate the structure for query answering. For instance, DAGs can express cause–effect relationship networks, and provide the mathematical operators to do prediction, diagnosis, planning, and also "what-if" analysis (in the form of counterfactuals" and conditioning), all within one mathematical and operational conceptual framework.

5.4. Addressing physical distribution

As stated in Section 2.1, in this paper we focus on the conceptual distribution of intelligent functionalities between architectural abstractions. In distributed intelligence though, it is essential to account for the physical distribution of components and their deployment onto computing nodes. Architectures based on AAs and DTs need to be mapped onto such a complex setting, and it is essential to tune the allocation of resources to keep the system operational.

In the context of DTs, cyber–physical awareness refers to the ability of a twin to continuously and accurately digitalise and represent its physical counterpart throughout its life cycle and evolution. This involves the seamless integration of models with the collection, processing, and analysis of data from the physical world. In addition, it includes the ability to measure the level of *entanglement* between the physical and digital counterparts, assessing the quality of their bidirectional interactions and the overall fidelity of the DT in accurately representing its physical counterpart [7,127]. Achieving and maintaining this awareness ensures a high-fidelity representation of the physical world, enabling dynamic adaptation and intelligent decision-making based on real-world conditions. This concept is critical for maintaining the integrity and effectiveness of DTs in complex and evolving operational environments.

Similarly, for AAs that need to work in close collaboration with DTs, it is important to deploy them taking into consideration the ability to communicate with one another using agent-to-agent communication, as well as the latency in both processing time and network communication [128] to perform actions on actuators through DTs.

5.5. Scalability

The deployment and operation of DTs can be resource-intensive, requiring significant computational power, storage, and energy. Even more so when we adopt the perspective endorsed in this paper that virtually every PA should be digitalised by its own DT. Managing resource consumption effectively is crucial to ensure the scalability and sustainability of DT implementations across different deployments and also across the Edge-to-Cloud compute continuum. Future research should focus on optimising the resource efficiency and scalability of DTs. This includes developing lightweight models and algorithms that reduce computational requirements, as well as exploring Edge computing solutions to offload DTs closer to the data source [129,130].

AAs may be deemed less worthy of these optimisations, at least in our vision, where they mostly map to the global applications' intelligent functionalities, not to the individual PAs composing the IoT deployment. However, scalability must not be overlooked at least for two reasons. First, AAs must anyway interact with possibly multiple DTs to deliver their functions, hence latency of communication at the very least must be kept as low as possible. This could translate to deploying and executing AAs in the Fog or even at the Edge, which already demands high scalability. Second, the micro-architecture **(E)** positions AAs as peers of DTs, to augment their capabilities. In this specific case, low-latency interactions and Edge deployment become crucial for AAs as well.

5.6. Systems of systems viewpoint

Besides raw "computational" scalability, in the sense of performance and execution efficiency, *conceptual scalability* of the design approach is also important: how easy it is to scale the design of any given sub-system to the overall System of Systems (SoS), for instance in terms of cognitive load imposed on the designers? The field of enterprise systems integration has ideas and proposals to offer in this sense, that have the benefit of being strongly rooted in real-world requirements and case studies. For instance, the FCBPSS architecture and design methodology has been proposed as a general purpose tool, cross-cutting application domains, and has been conceptually validated in the manufacturing domain [114]. Such a domain has strong bonds with CPSs and IoT, as often monitoring extends to the products to be assembled and control includes whole production pipelines or even supply chains—crossing the boundaries between multiple enterprises. There, design of the SoS is envisioned to unfold along 4 main steps [131]: first, defining the requirements model to specify, in turn, *F*unctions and their *C*ontext; second, design the *B*ehaviour and *S*tate of SoS the *C*omponents that ought to deliver such Functions; then, synthesise the operating *P*rinciples that should drive the overall SoS operations; finally, give *S*tructure to the Components (and their relationships) by introducing infrastructural constraints and/or desiderata (there including network concerns).

Although in this paper we took a different, somewhat complementary route in defining system architectures (bottom-up instead of top-down), there are similarities between the two approaches that are worth to be investigated in future work. For instance, the μ-arch that we propose lend themselves easily to recursive composition from individual components (e.g. predictive maintenance of some production appliance) to whole systems (e.g. shop-floor scheduling) and SoS as well (e.g. supply chain monitoring and control). The application of our proposed design principles and μ-arch both as part of a FCBPSS architecture design, and as a comparison to it surely is a promising direction for further work, that has the potential to further reduce reinventing the wheel in systems design.

5.7. A full-fledged methodology for DI in IoT systems

Software engineering is a complex activity that has many phases. The hardest ones often refer to the conversion of functional, technical, and non-functional requisites or desiderata into a system specification. Software methodologies exist to provide a structured way to perform this process, providing methods to speak with stakeholders, formalise requirements, and map them to the appropriate components of a software system.

Any software methodology is tightly linked to the main abstractions used to describe the system: for a desktop application we may consider objects and classes, for a Web service we may consider APIs, whereas for a user interface we may consider graphical components and event loops. With the proposals for the abstractions of AA and DTs as characterised in this paper, we believe that it might be extremely important to devise methodologies that are tailored to these abstractions for the engineering of intelligent IoT systems and applications.

As these systems have an important connection to the real world and are usually applied to sectors that involve different expertise, we turn to Domain-Driven Design [132] as a starting methodology to extract the domain knowledge from the involved experts and understand the requirements. DDD though has usually been paired with the design of microservice applications, and as such may need to be evolved to provide methods targeting directly intelligent and autonomous behaviour. We believe that an extension of such methodologies will be a much-needed step toward the realisation of a true *intelligence continuum* in IoT systems and that the abstractions and criteria provided in this paper can lay the ground for the definition of such methodologies.

6. Conclusion & outlook

In this paper, we analysed the existing literature about using AAs and DTs to deliver intelligence in IoT systems and applications. Based on this, we categorised the intelligent functionalities mentioned in the literature in classes, that we then take as a reference for proposing design principles. These design principles are meant to help software designers in assessing which abstraction is best suited to encapsulate a given intelligent functionality. Finally, we proposed a set of micro-architectures, stemming from such principles, that can be used by system designers to create the overall system architecture as a composition of modular pieces. These proposals are explicitly meant to assist IoT system designers in attacking the complexity of modern IoT scenarios and in navigating the abundance of technologies and mechanisms to realise them in a principled way.

We are fully aware that the challenges to fully realise this vision of an intelligence continuum in IoT systems are many, but we hope that our contribution may shed light on how AAs and DTs may contribute to the vision, and are confident that the present contribution could pave the way toward new research efforts integrating AAs and DTs within the framework we laid out.

CRediT authorship contribution statement

Samuele Burattini: Writing – original draft, Writing – review & editing, Methodology, Conceptualization. **Stefano Mariani:** Writing – original draft, Writing – review & editing, Supervision, Methodology, Conceptualization. **Sara Montagna:** Writing – original draft, Writing – review & editing, Conceptualization. **Marco Picone:** Writing – original draft, Writing – review & editing, Conceptualization. **Alessandro Ricci:** Writing – original draft, Methodology, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work is supported by the project "Digital Twins Ecosystems for the clinical, strategic and process governance in Healthcare" funded by the European Union - NextGenerationEU and Azienda Unità Sanitaria Locale (AUSL) della Romagna through the Italian "National Recovery and Resilience Plan, Italy" (PNRR) Mission 4, Component 2, Investment 3.3 (DM 352/2022), and by the European Union - NextGenerationEU - under PRIN 2022 project TWINKLE (project ID: 20223N7WCJ and CUP: E53D23007770001).

Data availability

No data was used for the research described in the article.

References

[1] Tariq A. A. Alsboui, Yongrui Qin, Richard Hill, Hussain Al-Aqrabi, Distributed intelligence in the Internet of Things: Challenges and opportunities, 2 (4), pp. 277. http://dx.doi.org/10.1007/S42979-021-00677-7.

[2] Daniel Rosendo, Alexandru Costan, Patrick Valduriez, Gabriel Antoniu, Distributed intelligence on the Edge-to-Cloud Continuum: A systematic literature review, 166, pp. 71–94, http://dx.doi.org/10.1016/J.JPDC.2022.04.004.

[3] Claudio Savaglio, Maria Ganzha, Marcin Paprzycki, Costin Bădică, Mirjana Ivanović, Giancarlo Fortino, Agent-based Internet of Things: State-of-the-art and research challenges, Futur. Gener. Comput. Syst. (ISSN: 0167-739X) 102 (2020) (2020) 1038–1053, http://dx.doi.org/10.1016/j.future.2019.09.016.

[4] Roberto Minerva, Gyu Myoung Lee, Noël Crespi, Digital twin in the IoT context: a survey on technical features, scenarios, and architectural models, 108, 10, pp. 1785–1824, http://dx.doi.org/10.1109/JPROC.2020.2998530.

[5] Petar Radanliev, Davi. De Roure, Razvan Nicolescu, Michael Huth, Omar Santos, Digital twins: artificial intelligence and the IoT cyber-physical systems in Industry 4.0. 6, 1 171–185, http://dx.doi.org/10.1007/S41315-021-00180-5.

[6] Payam M. Barnaghi, Amit P. Sheth, Cory A. Henson, From data to actionable knowledge: Big data challenges in the web of things, 28 (6) pp. 6–11 http://dx.doi.org/10.1109/MIS.2013.142.

[7] Roberto Minerva, Gyu Myoung Lee, Noël Crespi, Digital twin in the IoT context: A survey on technical features, scenarios, and architectural models, Proc. IEEE 108 (10) (2020) 1785–1824, 2020.

[8] Stefano Mariani, Marco Picone, Alessandro Ricci, About digital twins, agents, and multiagent systems: A cross-fertilisation journey, in: Francisco S. Melo, Fei Fang (Eds.), Autonomous Agents and Multiagent Systems. Best and Visionary Papers - AAMAS 2022 Workshops, Virtual Event, May 9-13, 2022, Revised Selected Papers, in: Lecture Notes in Computer Science, Vol. 13441, Springer, 2022, pp. 114–129, http://dx.doi.org/10.1007/978-3-031-20179-0_8.

[9] Christina Latsou, Maryam Farsi, John Ahmet Erkoyuncu, Geoffrey Morris, Digital twin integration in multi-agent cyber physical manufacturing systems, IFAC- Pap. 54 (1) (2021) 811–816, http://dx.doi.org/10.1016/j.ifacol.2021.08.096, 2021, Cited by: 3; All Open Access, Bronze Open Access, Green Open Access.

[10] Stefano Mariani, Marco Picone, Alessandro Ricci, Towards developing digital twin enabled multi-agent systems, in: Andrei Ciortea, Mehdi Dastani, Jieting Luo (Eds.), Engineering Multi-Agent Systems - 11th International Workshop, EMAS 2023, London, UK, May 29-30, 2023, Revised Selected Papers, in: Lecture Notes in Computer Science, vol. 14378, Springer, 2023, pp. 178–187, http://dx.doi.org/10.1007/978-3-031-48539-8_12.

[11] Daniel Rosendo, Alexandru Costan, Patrick Valduriez, Gabriel Antoniu, Distributed intelligence on the edge-to-cloud continuum: A systematic literature review, J. Parallel Distrib. Comput. 166 (2022) (2022) 71–94.

[12] Claudio Savaglio, Giancarlo Fortino, Mengchu Zhou, Towards interoperable, cognitive and autonomic IoT systems: An agent-based approach, in: 2016 IEEE 3rd World Forum on Internet of Things, WF-IoT, 2016, pp. 58–63, http://dx.doi.org/10.1109/WF-IoT.2016.7845459.

[13] Andrei Ciortea, Simon Mayer, Florian Michahelles, Repurposing manufacturing lines on the fly with multi-agent systems for the web of things, in: Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, (Stockholm, Sweden), AAMAS '18, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2018, pp. 813–822.

[14] Stefan Boschert, Roland Rosen, Digital twin—the simulation aspect, in: Mechatronic Futures, Springer, 2016, pp. 59–74.

[15] Haya Elayan, Moayad Aloqaily, Mohsen Guizani, Digital twin for intelligent context-aware IoT healthcare systems, IEEE Internet Things J. 8 (23) (2021) 16749–16757, http://dx.doi.org/10.1109/JIOT.2021.3051158, 2021.

[16] Thomas Clemen, Nima Ahmady-Moghaddam, Ulfia A. Lenfers, Florian Ocker, Daniel Osterholz, Jonathan Ströbele, Daniel Glake, Multi-agent systems and digital twins for smarter cities, in: Philippe J. Giabbanelli (Ed.), SIGSIM-PADS '21: SIGSIM Conference on Principles of Advanced Discrete Simulation, Virtual Event, USA, 31 May - 2 June, 2021, ACM, 2021, pp. 45–55, http://dx.doi.org/10.1145/3437959.3459254.

[17] Elena Pretel, Alejandro Moya, Elena Navarro, Víctor López-Jaquero, Pascual González, Analysing the synergies between Multi-agent Systems and Digital Twins: A systematic literature review. 174, 107503. http://dx.doi.org/10.1016/J.INFSOF.2024.107503.

[18] Alessandro Ricci, Angelo Croatti, Stefano Mariani, Sara Montagna, Marco Picone, Web of digital twins, ACM Trans. Internet Technol. (ISSN: 1533-5399) (2021) http://dx.doi.org/10.1145/3507909, dec2021. Just Accepted.

[19] Xabier Larrucea, Annie Combelles, John M. Favaro, Kunal Taneja, Software engineering for the Internet of Things, 34 (1) pp. 24–28, http://dx.doi.org/10.1109/MS.2017.28.

[20] Mahdi Fahmideh, Aakash Ahmad, Ali Behnaz, John Grundy, Willy Susilo, Software engineering for Internet of Things: The practitioners' perspective, 48 (8), pp. 2857–2878, http://dx.doi.org/10.1109/TSE.2021.3070692.

[21] Lynne E. Parker, Distributed intelligence: Overview of the field and its application in multi-robot systems, in: Tim Finin, Lalana Kagal, Elisa F. Kendall, Jason H. Li, Margaret Lyell, Walt Truszkowski (Eds.), Regarding the Intelligence in Distributed Intelligent Systems, Papers from the 2007 AAAI Fall Symposium, Arlington, Virginia, USA, November 9-11, 2007, in: AAAI Technical Report, vol. FS-07-06, AAAI Press, 2007, pp. 1–6, https://www.aaai.org/Library/Symposia/Fall/2007/fs07-06-002.php.

[22] Xiaolan Liu, Jiadong Yu, Yuanwei Liu, Yue Gao, Toktam Mahmoodi, Sangarapillai Lambotharan, Danny Hin-Kwok Tsang, Distributed intelligence in wireless networks, IEEE Open J. Commun. Soc 4 (2023) (2023) 1001–1039, http://dx.doi.org/10.1109/OJCOMS.2023.3265425.

[23] Swaminathan Gopalswamy, Sivakumar Rathinam, Infrastructure enabled autonomy: A distributed intelligence architecture for autonomous vehicles, in: 2018 IEEE Intelligent Vehicles Symposium, IV 2018, Changshu, Suzhou, China, June 26-30, 2018, IEEE, 2018, pp. 986–992, http://dx.doi.org/10.1109/IVS.2018.8500436.

[24] Schahram Dustdar, Victor Casamayor-Pujol, Praveen Kumar Donta, On distributed computing continuum systems, IEEE Trans. Knowl. Data Eng. 35 (4) (2023) 4092–4105, http://dx.doi.org/10.1109/TKDE.2022.3142856, 2023.

[25] Luiz Bittencourt, Roger Immich, Rizos Sakellariou, Nelson Fonseca, Edmundo Madeira, Marilia Curado, Leandro Villas, Luiz DaSilva, Craig Lee, Omer Rana, The Internet of Things, fog and cloud continuum: integration and challenges, Internet of Things 3 (2018) (2018) 134–155.

[26] Pierre Bourque, Robert Dupuis, Alain Abran, James W. Moore, Leonard L. Tripp, Sybille Wolff, Fundamental principles of software engineering - a journey, 62 (1) pp. 59–70, http://dx.doi.org/10.1016/S0164-1212(01)00136-4.

[27] V.I. Gorodetsky, S.S. Kozhevnikov, D. Novichkov, P.O. Skobelev, The framework for designing autonomous cyber-physical multi-agent systems for adaptive resource management, in: Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), in: LNAI, vol. 11710, 2019, pp. 52–64, http://dx.doi.org/10.1007/978-3-030-27878-6_5, 2019, Cited by: 5.

[28] Tariq Alsboui, Yongrui Qin, Richard Hill, Hussain Al-Aqrabi, Distributed intelligence in the Internet of Things: challenges and opportunities, SN Comput. Sci. 2 (4) (2021) 277, 2021.

[29] Michael J. Wooldridge, Nicholas R. Jennings, Intelligent agents: theory and practice, Knowl. Eng. Rev. 10 (2) (1995) 115–152, (1995).

[30] Ghezlan. Halhoul Merabet, Mohammed Essaaidi, Hanaa Talei, Mohame. Riduan Abid, Nacer Khalil, Mohcine Madkour, Driss Benhaddou, Applications of multi-agent systems in smart grids: A survey, in: 4th International Conference on Multimedia Computing and Systems, ICMCS 2014, Marrakech, Morocco, April 14-16, 2014, IEEE, 2014, pp. 1088–1094, http://dx.doi.org/10.1109/ICMCS.2014.6911384.

[31] Emilio Sulis, Stefano Mariani, Sara Montagna, A survey on agents applications in healthcare: Opportunities, challenges trends, Comput. Methods Programs Biomed. 236 (2023) (2023) 107525, http://dx.doi.org/10.1016/J.CMPB.2023.107525.

[32] Weiming Shen, Distributed manufacturing scheduling using intelligent agents, IEEE Intell. Syst. 17 (1) (2002) 88–94, http://dx.doi.org/10.1109/5254.988492, 2002.

[33] Rafael H. Bordini, Lars Braubach, Mehdi Dastani, Amal E. Fallah Seghrouchni, Jorge J. Gómez-Sanz, João Leite, Gregory M.P. O'Hare, Alexander Pokahr, Alessandro Ricci, A survey of programming languages and platforms for multi-agent systems, Inform. (Slovenia) 30 (1) (2006) 33–44, 2006, http://www.informatica.si/index.php/informatica/article/view/71.

[34] Costin Badica, Zoran Budimac, Hans-Dieter Burkhard, Mirjana Ivanovic, Software agents: Languages, tools, platforms, Comput. Sci. Inf. Syst. 8 (2) (2011) 255–298, http://dx.doi.org/10.2298/CSIS110214013B, 2011.

[35] Joanna Bryson, Cross-paradigm analysis of autonomous agent architecture, J. Exp. Theor. Artif. Intell. 12 (2) (2000) 165–189, http://dx.doi.org/10.1080/095281300409829, (2000).

[36] Gerhard Weiss (Ed.), Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence, MIT Press, Cambridge, MA, ISBN: 978-0262731317, 1999.

[37] Rafael H. Bordini, Amal El Fallah Seghrouchni, Koen Hindriks, Brian Logan, Alessandro Ricci, Agent programming in the cognitive era, Auton. Agents Multi- Agent Syst. 34 (2) (2020) 37, http://dx.doi.org/10.1007/s10458-020-09453-y, 2020,

[38] Rosario Girardi, Adriana Leite, A survey on software agent architectures, IEEE Intell. Inform. Bull. 14 (1) (2013) 8–20, 2013, http://www.comp.hkbu.edu.hk/%7Eiib/2013/Dec/article2/iib_vol14no1_article2.pdf.

[39] Jörg P. Müller, Klaus Fischer, Application impact of multi-agent systems and technologies: A survey, in: Arnon Sturm Onn Shehory (Ed.), Agent-Oriented Software Engineering - Reflections on Architectures, Methodologies, Languages, and Frameworks, Springer, 2014, pp. 27–53, http://dx.doi.org/10.1007/978-3-642-54432-3_3.

[40] Abid Khan, Sadia Din, Gwanggil Jeon, Francesco Piccialli, Lucy with agents in the sky: Trustworthiness of cloud storage for industrial Internet of Things, IEEE Trans. Ind. Inform. 17 (2) (2021) 953–960, http://dx.doi.org/10.1109/TII.2020.2974493, 2021.

[41] Munindar P. Singh, Amit K. Chopra, The Internet of Things and multiagent systems: Decentralized intelligence in distributed computing, in: Ling Liu Kisung Lee (Ed.), 2017 IEEE 37th International Conference on Distributed Computing Systems, ICDCS (2017), IEEE Computer Society, 2017, pp. 1738–1747, http://dx.doi.org/10.1109/ICDCS.2017.304.

[42] Andrea Omicini, Roberta Calegari, Injecting (micro)intelligence in the IoT: Logic-based approaches for (M)MAS, in: Donghui Lin, Toru Ishida, Franco Zambonelli, Itsuki Noda (Eds.), Massively Multi-Agent Systems II, Springer International Publishing, Cham, ISBN: 978-3-030-20937-7, 2019, pp. 21–35.

[43] Carmelo Fabio Longo, Francesco Longo, Corrado Santoro, Caspar: towards decision making helpers agents for IoT, based on natural language and first order logic reasoning, Eng. Appl. Artif. Intell. (ISSN: 0952-1976) 104 (2021) (2021) 104269, http://dx.doi.org/10.1016/j.engappai.2021.104269.

[44] Paulo Leitão, Stamatis Karnouskos, Luis Ribeiro, Jay Lee, Thomas I. Strasser, Armando W. Colombo, Smart agents in industrial cyber-physical systems, Proc. IEEE 104 (5) (2016) 1086–1101, http://dx.doi.org/10.1109/JPROC.2016.2521931, 2016.

[45] Franco Cicirelli, Giancarlo Fortino, Andrea Giordano, Antonio Guerrieri, Giandomenico Spezzano, Andrea Vinci, On the design of smart homes: A framework for activity recognition in home environment, J. Med. Syst. 40 (9) (2016) 200:1–200:17, http://dx.doi.org/10.1007/S10916-016-0549-7, (2016).

[46] Francesco Lanza, Valeria Seidita, Antonio Chella, Agents and robots for collaborating and supporting physicians in healthcare scenarios, J. Biomed. Inform. (ISSN: 1532-0464) 108 (2020) (2020) 103483, http://dx.doi.org/10.1016/j.jbi.2020.103483.

[47] Yingfeng Zhang, Cheng Qian, Jingxiang Lv, Ying Liu, Agent and cyber-physical system based self-organizing and self-adaptive intelligent shopfloor, IEEE Trans. Ind. Informatics 13 (2) (2017) 737–747, http://dx.doi.org/10.1109/TII.2016.2618892, (2017).

[48] Hao Tang, Di Li, Shiyong Wang, Zhijie Dong, CASOA: An architecture for agent-based manufacturing system in the context of industry 4.0, IEEE Access 6 (2018) (2018) 12746–12754, http://dx.doi.org/10.1109/ACCESS.2017.2758160.

[49] Julia Rosenberger, Michael Urlaub, Felix Rauterberg, Tina Lutz, Andreas Selig, Michael Bühren, Dieter Schramm, Deep reinforcement learning multi-agent system for resource allocation in industrial Internet of Things, Sensors (ISSN: 1424-8220) 22 (11) (2022) 2022, http://dx.doi.org/10.3390/s22114099.

[50] Jiachen Yang, Jipeng Zhang, Huihui Wang, Urban traffic control in software defined Internet of Things via a multi-agent deep reinforcement learning approach, IEEE Trans. Intell. Transp. Syst. vol. 22 (6) (2021) 3742–3754, http://dx.doi.org/10.1109/TITS.2020.3023788, 2021.

[51] Lei Lei, Yue Tan, Kan Zheng, Shiwen Liu, Kuan Zhang, Xuemin Shen, Deep reinforcement learning for autonomous Internet of Things: Model, applications and challenges, IEEE Commun. Surv. Tutorials 22 (3) (2020) 1722–1760, http://dx.doi.org/10.1109/COMST.2020.2988367, 2020.

[52] Tong Zhou, Dunbing Tang, Haihua Zhu, Zequn Zhang, Multi-agent reinforcement learning for online scheduling in smart factories, Robot. Comput. Integr. Manuf. vol. 72 (2021) (2021) 102202, http://dx.doi.org/10.1016/J.RCIM.2021.102202.

[53] Ying Liu, Lei Liu, Wei-Peng Chen, Intelligent traffic light control using distributed multi-agent Q learning, in: 20th IEEE International Conference on Intelligent Transportation Systems, ITSC 2017, Yokohama, Japan, October 16-19, 2017, IEEE, 2017, pp. 1–8, http://dx.doi.org/10.1109/ITSC.2017.8317730.

[54] Long Zhang, Xiaozheng Ma, Zirui Zhuang, Haitao Xu, Vishal Sharma, Zhu Han, Q-learning aided intelligent routing with maximum utility in cognitive UAV swarm for emergency communications, IEEE Trans. Veh. Technol 72 (3) (2023a) 3707–3723, http://dx.doi.org/10.1109/TVT.2022.3221538, (2023).

[55] Samreen Laghari, Muaz A. Niazi, Modeling the Internet of Things, self-organizing and other complex adaptive communication networks: A cognitive agent-based computing approach, PLoS One 11 (1) (2016) 012016, http://dx.doi.org/10.1371/journal.pone.0146760, 1–26.

[56] M Majid Butt, Indrakshi Dey, Merim Dzaferagic, Maria Murphy, Nicholas Kaminski, Nicola Marchetti, Agent-based modeling for distributed decision support in an IoT network, IEEE Internet Things J. 7 (8) (2020) 6919–6931, http://dx.doi.org/10.1109/JIOT.2020.2976802, 2020.

[57] Ashish Kumar Shakya, Gopinatha Pillai, Sohom Chakrabarty, Reinforcement learning algorithms: A brief survey, 231, pp. 120495, http://dx.doi.org/10.1016/J.ESWA.2023.120495.

[58] Stefan Mihai, Mahnoor Yaqoob, Dang V. Hung, William Davis, Praveer Towakel, Mohsin Raza, Mehmet Karamanoglu, Balbir Barn, Dattaprasad Shetve, Raja V. Prasad, et al., Digital twins: A survey on enabling technologies, challenges, trends and future prospects, IEEE Commun. Surv. & Tutorials (2022) 2022.

[59] Michael Batty, Digital twins, 2018, pp. 817–820.

[60] Fei Tao, Qinglin Qi, Make more digital twins, Nat. (ISSN: 14764687) 573 (7775) (2019) 490–491, http://dx.doi.org/10.1038/d41586-019-02849-1, (2019).

[61] Karl Hribernik, Giacomo Cabri, Federica Mandreoli, Gregoris Mentzas, Autonomous, context-aware, adaptive digital twins - state of the art and roadmap, Comput. Ind. 133 (2021) (2021) 103508, http://dx.doi.org/10.1016/j.compind.2021.103508.

[62] Daniel Lehner, Jérôme Pfeiffer, Erik-Felix Tinsel, Matthias Milan Strljic, Sabine Sint, Michael Vierhauser, Andreas Wortmann, Manuel Wimmer, Digital twin platforms: Requirements, capabilities, and future prospects, IEEE Softw. 39 (2) (2022) 53–61, http://dx.doi.org/10.1109/MS.2021.3133795, 2022.

[63] Giancarlo Fortino, Claudio Savaglio, Integration of Digital Twins & Internet of Things, Springer International Publishing, ISBN: 978-3-031-21343-4, pp. 205–225, http://dx.doi.org/10.1007/978-3-031-21343-4_8.

[64] Akram Hakiri, Aniruddha Gokhale, Sadok Ben Yahia, Nedra Mellouli, A comprehensive survey on digital twin for future networks and emerging Internet of Things industry, 244, 110350, http://dx.doi.org/10.1016/J.COMNET.2024.110350.

[65] ETSI Specialist Task Forces (STF) 628, SmartM2M; digital twins communication requirements, 2024, TS TS-103-845-V1.1.1. European Telecommunications Standards Institute (ETSI).

[66] ISO/IEC 30173:20232023, Digital Twin – Concepts and terminology, Standard. International Organization for Standardization.

[67] Enxhi Ferko, Alessio Bucaioni, Patrizio Pelliccione, Moris Behnam, Standardisation in digital twin architectures in manufacturing, in: 2023 IEEE 20th International Conference on Software Architecture (ICSA), 2023, pp. 70–81, http://dx.doi.org/10.1109/ICSA56044.2023.00015.

[68] Prasad Talasila, Cláudio Gomes, Lars B. Vosteen, Hannes Iven, Martin Leucker, Santiago Gil, Peter H. Mikkelsen, Eduard Kamburjan, Peter G. Larsen, Composable digital twins on digital twin as a service platform, SIMULATION 2024 (2024) 00375497241298653, http://dx.doi.org/10.1177/00375497241298653.

[69] Antonello Barbone, Samuele Burattini, Matteo Martinelli, Marco Picone, Alessandro Ricci, Antonio Virdis, Digital twin continuum: a key enabler for pervasive cyber-physical environments, in: 2024 33rd International Conference on Computer Communications and Networks, ICCCN, 2024, pp. 1–9, http://dx.doi.org/10.1109/ICCCN61486.2024.10637565.

[70] Shohin Aheleroff, Xun Xu, Ray Y. Zhong, Yuqian Lu, Digital Twin as a Service (DTaaS) in Industry 4.0: An architecture reference model, Adv. Eng. Inform. (ISSN: 1474-0346) 47 (2021) (2021) 101225, http://dx.doi.org/10.1016/j.aei.2020.101225.

[71] Jingxi Zhang, Carsten Ellwein, Malte Heithoff, Judith Michael, Andreas Wortmann, Digital twin and the asset administration shell: An analysis of the three types of AASs and their feasibility for digital twin engineering, Softw. Syst. Model. (ISSN: 1619-1374) (2025) (2025) http://dx.doi.org/10.1007/s10270-024-01255-0.

[72] Cheng Zhou, Hongwei Yang, Xiaodong Duan, Diego Lopez, Antonio Pastor, Qin Wu, Mohamed Boucadair, Christian Jacquenet, Digital twin network: Concepts and reference architecture, 2022, Internet-Draft draft-irtf-nmrg-network-digital-twin-arch-01. Internet Engineering Task Force. https://datatracker.ietf.org/doc/draft-irtf-nmrg-network-digital-twin-arch/01/ Work in Progress.

[73] Jinzhi Lu Xiaoche. Zheng, Dimitris Kiritsis, The emergence of cognitive digital twin: vision, challenges and opportunities, Int. J. Prod. Res. 60 (24) (2022) 7610–7632, http://dx.doi.org/10.1080/00207543.2021.2014591, 2022.

[74] Peter Gorm Larsen, John Fitzgerald, Cláudio Gomes, Engineering Digital Twins for Cyber-Physical Systems, Springer International Publishing, ISBN: 9783031667190, 2024, pp. 3–17, http://dx.doi.org/10.1007/978-3-031-66719-0_1.

[75] Zhuming Bi, Chris W.J. Zhang, Chong Wu, Ling Li, New digital triad (DT-II) concept for lifecycle information integration of sustainable manufacturing systems, J. Ind. Inf. Integr. 26 (2022) (2022) 100316, http://dx.doi.org/10.1016/J.JII.2021.100316.

[76] Jinfeng Liu, Honggen Zhou, Xiaojun Liu, Guizhong Tian, Mingfang Wu, Liping Cao, Wei Wang, Dynamic evaluation method of machining process planning based on digital twin, IEEE Access 7 (2019) (2019) 19312–19323.

[77] Chunhua Hu, Weicun Fan, Elan Zeng, Zhi Hang, Fan Wang, Lianyong Qi, Md. Zakirul Alam Bhuiyan, Digital twin-assisted real-time traffic data prediction method for 5G-enabled internet of vehicles, 18 (4), pp. 2811–2819, http://dx.doi.org/10.1109/TII.2021.3083596.

[78] Qianzhe Qiao, Jinjiang Wang, Lunkuan Ye, Robert X. Gao, Digital twin for machining tool condition prediction. 81, 2019, pp. 1388–1393, http://dx.doi.org/10.1016/j.procir.2019.04.049, 2212-8271, 52nd CIRP Conference on Manufacturing Systems (CMS), Ljubljana, Slovenia, June 12-14.

[79] Weiwei Qian, Yu Guo, Hao Zhang, Shaohua Huang, Litong Zhang, Hailang Zhou, Weiguang Fang, Shanshan Zha, Digital twin driven production progress prediction for discrete manufacturing workshop, 80, 102456, http://dx.doi.org/10.1016/J.RCIM.2022.102456.

[80] Long Zhang, Han Wang, Hongmei Xue, Hongliang Zhang, Qilie Liu, Dusit Niyato, Zhu Han, Digital twin-assisted edge computation offloading in industrial Internet of Things with NOMA, IEEE Trans. Veh. Technol 72 (9) (2023b) 11935–11950, http://dx.doi.org/10.1109/TVT.2023.3270859, (2023).

[81] Mansoor Ali, Georges Kaddoum, Wen-Tai Li, Chau Yuen, Muhammad Tariq, H. Vincent Poor, A smart digital twin enabled security framework for vehicle-to-grid cyber-physical systems, IEEE Trans. Inf. Forensics Secur. 18 (2023) (2023) 5258–5271, http://dx.doi.org/10.1109/TIFS.2023.3305916.

[82] Manuel S. Müller, Nasser Jazdi, Michael Weyrich, Self-improving models for the intelligent digital twin: Towards closing the reality-to-simulation gap, IFAC- Pap. (ISSN: 2405-8963) 55 (2) (2022) 126–131, http://dx.doi.org/10.1016/j.ifacol.2022.04.181, 2022, 14th IFAC Workshop on Intelligent Manufacturing Systems IMS 2022.

[83] Yan Xu, Yanming Sun, Xiaolong Liu, Yonghua Zheng, A digital-twin-assisted fault diagnosis using deep transfer learning, IEEE Access 7 (2019) (2019) 19990–19999.

[84] Jack Sleuters, Yonghui Li, Jacques Verriet, Marina Velikova, Richard Doornbos, A digital twin method for automated behavior analysis of large-scale distributed IoT systems, in: 2019 14th Annual Conference System of Systems Engineering, SoSE, IEEE, 2019, pp. 7–12.

[85] Yuchong Qian, Jiawei Leng, Kai Zhou, Yuxuan Liu, How to measure and control indoor air quality based on intelligent digital twin platforms: A case study in China, Build. Environ. (ISSN: 0360-1323) 253 (2024) (2024) 111349, http://dx.doi.org/10.1016/j.buildenv.2024.111349.

[86] Longyu Zhou, Supeng Leng, Tony Q.S. Quek, Hierarchical digital-twin-enhanced cooperative sensing for UAV swarms, IEEE Internet Things J 11 (20) (2024) 33204–33216, http://dx.doi.org/10.1109/JIOT.2024.3428476, (2024).

[87] Vladimir Kuts, Tauno Otto, Toivo Tähemaa, Yevhen Bondarenko, Digital twin based synchronised control and simulation of the industrial robotic cell using virtual reality, J. Mach. Eng. 19 (2019) 022019, 128–144.

[88] Wladimir Hofmann, Fredrik Branding, Implementation of an IoT- and cloud-based digital twin for real-time decision support in port operations, (ISSN: 2405-8963) 52 (13) pp. 2104–2109, 9th IFAC Conference on Manufacturing Modelling, Management and Control MIM 2019, http://dx.doi.org/10.1016/j.ifacol.2019.11.516.

[89] Behin Elahi, Sadeg. Amiri Tokaldany, Chapter 15 - application of Internet of Things-aided simulation and digital twin technology in smart manufacturing, in: Mangey Ram (Ed.), Advances in Mathematics for Industry 4.0, Academic Press, ISBN: 978-0-12-818906-1, pp. 335–359, http://dx.doi.org/10.1016/B978-0-12-818906-1.00015-2.

[90] Abdallah Karakra, Franck Fontanili, Elyes Lamine, Jacques Lamothe, HospiT'Win: A predictive simulation-based digital twin for patients pathways in hospital, in: 2019 IEEE EMBS International Conference on Biomedical & Health Informatics, BHI 2019, Chicago, IL, USA, May 19-22, 2019, IEEE, 2019, pp. 1–4, http://dx.doi.org/10.1109/BHI.2019.8834534.

[91] William Danilczyk, Yan Sun, Haibo He, ANGEL: An intelligent digital twin framework for microgrid security, in: 2019 North American Power Symposium, NAPS, 2019, pp. 1–6, http://dx.doi.org/10.1109/NAPS46351.2019.9000371.

[92] Huan Wang, Xiaoqiang Di, Yan Wang, Bin Ren, Ge Gao, Junyi Deng, An intelligent digital twin method based on spatio-temporal feature fusion for IoT attack behavior identification, IEEE J. Sel. Areas Commun. 41 (11) (2023) 3561–3572, http://dx.doi.org/10.1109/JSAC.2023.3310091, 2023.

[93] Min Deng, Carol C. Menassa, Vineet R. Kamat, From BIM to digital twins: A systematic review of the evolution of intelligent building representations in the AEC-FM industry, J. Inf. Technol. Constr. 26 (2021) 2021.

[94] Christos Pylianidis, Sjoukje Osinga, Ioannis N. Athanasiadis, Introducing digital twins to agriculture, Comput. Electron. Agric. 184 (2021) (2021) 105942.

[95] Mamoun Alazab, Latif U. Khan, Srinivas Koppu, Swarna Priya Ramu, M. Iyapparaja, Parimala Boobalan, Thar Baker, Praveen Kumar Reddy Maddikunta, Thippa Reddy Gadekallu, Ahamed Aljuhani, Digital twins for healthcare 4.0-recent advances, architecture, and open challenges, IEEE Consum. Electron. Mag. (2022) 2022.

[96] Roberto Minerva, Noel Crespi, Reza Farahbakhsh, Faraz M. Awan, Artificial Intelligence and the Digital Twin: An Essential Combination, Springer International Publishing, Cham, ISBN: 978-3-031-21343-4, 2023, pp. 299–336, http://dx.doi.org/10.1007/978-3-031-21343-4_12.

[97] Maria G. Juarez, Vicente J. Botti, Adriana S. Giret, Digital twins: Review and challenges, J. Comput. Inf. Sci. Eng. (ISSN: 1530-9827) 21 (3) (2021) 042021, http://dx.doi.org/10.1115/1.4050244, https://asmedigitalcollection.asme.org/computingengineering/article-pdf/21/3/030802/6696358/jcise_21_3_030802.pdf, 030802.

[98] Yogeswaranathan Kalyani, Rem W. Collier, The role of multi-agents in digital twin implementation: Short survey, 57 (3) pp. 72:1–72:15, http://dx.doi.org/10.1145/3697350.

[99] Qingwei Nie, Dunbing Tang, Haihua Zhu, Hongwei Sun, A multi-agent and Internet of Things framework of digital twin for optimized manufacturing control, 35 (10-11), pp. 1205–1226, http://dx.doi.org/10.1080/0951192X.2021.2004619.

[100] Wenjun Xu, Hang Yang, Zhenrui Ji, Mengyuan Ba, Cognitive digital twin-enabled multi-robot collaborative manufacturing: Framework and approaches, 194, pp. 110418, http://dx.doi.org/10.1016/J.CIE.2024.110418.

[101] Michael E. Bratman, David J. Israel, Martha E. Pollack, Plans and resource-bounded practical reasoning, 4 pp. 349–355, http://dx.doi.org/10.1111/J.1467-8640.1988.TB00284.X.

[102] John-Jules Charles Meyer, Wiebe van der Hoek, Epistemic logic for AI and computer science, in: Cambridge tracts in theoretical computer science, vol. 41, Cambridge University Press, ISBN: 978-0-521-46014-9.

[103] Ji-Young Oh, Claudia Raibulet, Joran Leest, Analysis of MAPE-K loop in self-adaptive systems for cloud, IoT and CPS, in: Javier Troya, Raffaela Mirandola, Elena Navarro, Andrea Delgado, Sergio Segura, Guadalupe Ortiz, Cesare Pautasso, Christian Zirpins, Pablo FernánDez, Antonio Ruiz-Cortés (Eds.), Service-Oriented Computing - ICSOC 2022 Workshops - ASOCA, AI-PA, FMCIoT, WESOACS 2022, Sevilla, Spain, November 29 - December 2, 2022 Proceedings, in: Lecture Notes in Computer Science, vol. 13821, Springer, 2022, pp. 130–141, http://dx.doi.org/10.1007/978-3-031-26507-5_11.

[104] A.E. Eiben, Evolutionary computing and autonomic computing: Shared problems, shared solutions? in: Öalp Babaoglu, Márk Jelasity, Alberto Montresor, Christof Fetzer, Stefano Leonardi, Aad P.A. van Moorsel, Maarten van Steen (Eds.), Self-Star Properties in Complex Information Systems, Conceptual and Practical Foundations [The Book Is a Result from a Workshop at Bertinoro, Italy, Summer 2004], in: Lecture Notes in Computer Science, vol. 3460, Springer, 2005, pp. 36–48, http://dx.doi.org/10.1007/11428589_3.

[105] Ömer Ibrahim Erduran, Machine learning for cognitive BDI agents: A compact survey, in: Proceedings of the 15th International Conference on Agents and Artificial Intelligence, ICAART 2023, Volume 1, Lisbon, Portugal, February 22-24, 2023, SCITEPRESS, 2023, pp. 257–268, http://dx.doi.org/10.5220/0011678100003393.

[106] Stefano Mariani, Franco Zambonelli, Learning stigmergic communication for self-organising coordination, in: IEEE International Conference on Autonomic Computing and Self-Organizing Systems, ACSOS 2023, Toronto, on, Canada, September 25-29, 2023, IEEE, 2023, pp. 47–56, http://dx.doi.org/10.1109/ACSOS58161.2023.00022.

[107] Stefano Mariani, Pasquale Roseti, Franco Zambonelli, Towards multi-agent learning of causal networks, in: Noa Agmon Bo an, Alessandro Ricci, William Yeoh (Eds.), Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2023, London, United Kingdom, 29 May 2023 - 2 June 2023, ACM, 2023, pp. 2807–2809, http://dx.doi.org/10.5555/3545946.3599085.

[108] Stuart J. Russell, Peter Norvig, Artificial Intelligence - a Modern Approach: The Intelligent Agent Book, Prentice Hall, ISBN: 978-0-13-103805-9, 1995, https://www.worldcat.org/oclc/31288015.

[109] N. Crespi, A.T. Drobot, R. Minerva, The Digital Twin, Springer International Publishing, ISBN: 9783031213434, 2023, https://books.google.it/books?id=lITCEAAAQBAJ.

[110] Somayeh Malakuti, Sten Grüner, Architectural aspects of digital twins in IIoT systems, in: Proceedings of the 12th European Conference on Software Architecture: Companion Proceedings, (Madrid, Spain), ECSA '18, Association for Computing Machinery, New York, NY, USA, 2018, 12, 2.

[111] Dominique Guinard, Vlad Trifa, Erik Wilde, A resource oriented architecture for the Web of Things, in: 2010 Internet of Things, IOT, IEEE, 2010, pp. 1–8.

[112] Jasmin Guth, Uwe Breitenbücher, Michael Falkenthal, Frank Leymann, Lukas Reinfurt, Comparison of IoT platform architectures: A field study based on a reference architecture, in: 2016 Cloudification of the Internet of Things, CIoT 2016, Paris, France, November 23-25, 2016, IEEE, 2016, pp. 1–6, http://dx.doi.org/10.1109/CIOT.2016.7872918.

[113] Everton Cavalcante, Marcelo Pitanga Alves, Thais Batista, Flavia Coimbra Delicato, Paulo F. Pires, An analysis of reference architectures for the Internet of Things, in: Proceedings of the 1st International Workshop on Exploring Component-Based Techniques for Constructing Reference Architectures, (Montrél, QC, Canada, 2015) (CobRA '15), Association for Computing Machinery, New York, NY, USA, ISBN: 9781450334457, pp. 13–16, http://dx.doi.org/10.1145/2755567.2755569.

[114] Junwei Wang, Hongfeng Wang, J.L Ding, Kazuo Furuta, Taro Kanno, Andrew W.H. Ip, Wenjun Zhang, On domain modelling of the service system with its application to enterprise information systems, Enterp. Inf. Syst. 10 (1) (2016) 1–16, http://dx.doi.org/10.1080/17517575.2013.810784, 2016.

[115] Pablo Pico-Valencia, Juan A. Holgado-Terriza, Agentification of the Internet of Things, A systematic literature review, 14, 10. http://dx.doi.org/10.1177/1550147718805945.

[116] Ahmad Alelaimat, Aditya Ghose, Hoa Khanh Dam, Abductive design of BDI agent-based digital twins of organizations, in: PRIMA 2020: Principles and Practice of Multi-Agent Systems - 23rd International Conference, in: LNCS, vol. 12568, Springer, 2020, pp. 377–385.

[117] Christian Stary, Digital twin generation: Re-conceptualizing agent systems for behavior-centered cyber-physical system development, Sensors (ISSN: 1424-8220) 21 (4) (2021) 2021.

[118] Hang Wan, Michael David, William Derigent, Modelling digital twins as a recursive multi-agent architecture: application to energy management of communicating materials, IFAC- Pap. (ISSN: 2405-8963) 54 (1) (2021) 880–885, http://dx.doi.org/10.1016/j.ifacol.2021.08.104, 2021.

[119] Davide Calvaresi, Mauro Marinoni, Arnon Sturm, Michael Schumacher, Giorgio Buttazzo, The challenge of real-time multi-agent systems for enabling IoT and CPS, in: Proceedings of the International Conference on Web Intelligence, (Leipzig, Germany), WI '17, Association for Computing Machinery, New York, NY, USA, ISBN: 9781450349512, 2017, pp. 356–364, http://dx.doi.org/10.1145/3106426.3106518.

[120] Wenjie Jia, Wei Wang, Zhenzu Zhang, From simple digital twin to complex digital twin Part I: A novel modeling method for multi-scale and multi-scenario digital twin, 53 pp. 101706. http://dx.doi.org/10.1016/J.AEI.2022.101706.

[121] Zainab Salih Ageed, Subhi R.M. Zeebaree, Mohammed A.M. Sadeeq, Maiwan Bahjat Abdulrazzaq, Baraa Wasfi Salim, Azar Abid Salih, Hajar Maseeh Yasin, Awder M. Ahmed, A state of art survey for intelligent energy monitoring systems, Asian J. Res. Comput. Sci. (2021) (2021).
[122] Wei Cai, Liang Wang, Li Li, Jun Xie, Shun Jia, Xugang Zhang, Zhidi Jiang, Kee hung Lai, A review on methods of energy performance improvement towards sustainable manufacturing from perspectives of energy monitoring, evaluation, optimization and benchmarking, Renew. Sustain. Energy Rev. 2022 (2022).
[123] Tanveer Hussain, Fath U. Min Ullah, Khan Muhammad, Seungmin Rho, Amin Ullah, Eenjun Hwang, Jihoon Moon, Sun. Wook Baik, Smart and intelligent energy monitoring systems: A comprehensive literature survey and future research guidelines, Int. J. Energy Res. 45 (2021) (2021) 3590–3614.
[124] Fabio Bellifemine, FIPA: a standard for agent interoperability, in: WOA 2000: Dagli Oggetti Agli Agenti. 1st AI*IA/TABOO Joint Workshop "from Objects To Agents": Evolutive Trends of Software Systems, 29-30 May 2000, Parma, Italy, Pitagora Editrice Bologna, 2000, p. 121.
[125] Anand S. Rao, Michael Georgeff, BDI agents: From theory to practice, in: ICMAS, 1995.
[126] Judea Pearl, Graphical models for probabilistic and causal reasoning, in: Teofilo F. Gonzalez, Jorge Diaz-Herrera, Allen Tucker (Eds.), Computing Handbook, Third Edition: Computer Science and Software Engineering, CRC Press, pp. 441–24.
[127] Paolo Bellavista, Nicola Bicocchi, Mattia Fogli, Carlo Giannelli, Marco Mamei, Marco Picone, ODTE: A metric for digital twin entanglement, IEEE Open J. Commun. Soc. 5 (2024) (2024) 2377–2390, http://dx.doi.org/10.1109/OJCOMS.2024.3385659.
[128] Davide Calvaresi, Yashin Dicente Cid, Mauro Marinoni, Aldo Franco Dragoni, Amro Najjar, Michael Schumacher, Real-time multi-agent systems: rationality, formal model, and empirical results, Auton. Agents Multi Agent Syst. 35 (1) (2021) 12, http://dx.doi.org/10.1007/S10458-020-09492-5, 2021.
[129] Claudio Cicconetti, Marco Conti, Andrea Passarella, Architecture and performance evaluation of distributed computation offloading in edge computing, Simul. Model. Pr. Theory (ISSN: 1569-190X) 101 (2020) 102007, http://dx.doi.org/10.1016/j.simpat.2019.102007, 2020, Modeling and Simulation of Fog Computing.
[130] Marco Picone, Marco Mamei, Franco Zambonelli, A flexible and modular architecture for edge digital twin: Implementation and evaluation, ACM Trans. Internet Things vol. 4 (1) (2023) 8, http://dx.doi.org/10.1145/3573206, feb2023, 32.
[131] Wenju. Chris Zhang, Junwei Wang, Design theory and methodology for enterprise systems, Enterp. Inf. Syst. 10 (3) (2016) 245–248, http://dx.doi.org/10.1080/17517575.2015.1080860, (2016).
[132] Eric Evans, Domain-Driven Design: Tackling Complexity in the Heart of Software, Addison-Wesley Professional, 2004.