Digital Twins and Intelligence: A Symbiotic Framework

By

Nan Zhang

A thesis submitted to
the University of Birmingham
for the degree of
DOCTOR OF PHILOSOPHY

School of Computer Science
College of Engineering and Physical Sciences
University of Birmingham
September 2023

ABSTRACT

In the evolving landscape of intelligent cyber-physical systems, the concept of Digital Twins (DTs) emerges as a pivotal paradigm. The DT represents a symbiotic relationship between a virtual model and its physical counterpart, offering insights and control for the physical system. When combining intelligence with DTs, research is still inadequate in considering intelligence manifested in both the real-world system and the DT. The foundational motivation stems from the increasing prevalence of autonomy and intelligence in modern systems, which often operate within intricate and dynamic environments. Systems in this context can be designed to be computationally self-aware, but may suffer from limited computational resources, which restricts their level of intelligence. DTs can offload their computational burden to offer more informed analysis to overcome the restriction. However, as self-aware systems accumulate knowledge and exhibit intelligence, the role of DTs in enhancing their capabilities becomes a compelling question. The central inquiry guiding this research is: How should an intelligent DT be designed to facilitate an intelligent system which is already endowed with computational self-awareness? This thesis proposes a novel notion of *mutual intelligence enrichment*, which enables runtime knowledge of the DT and the system to be utilised by each other to boost more adaptive intelligent behaviours. This thesis proposes a novel holistic reference architecture to address the problem with mechanisms in different dimensions: cognitive capabilities of the DT, physical-to-virtual model update, and virtual-to-physical system adaptation. This reference architecture leverages principles from self-awareness and Dynamic Data-Driven Applications Systems (DDDAS) to address

the challenges of model equivalence maintenance, adaptive runtime trade-off analysis, and explainability for human-in-the-loop. The major benefit is that by levering the architecture, equivalence and adaptation can be conducted from a knowledge perspective with minimum human intervention. Also, it can enable explainability if humans are involved in the decision loop. Evaluation in different application domains shows the efficiency and validity of the proposed approaches.

ACKNOWLEDGMENTS

First and foremost, I would like to express my gratitude to my supervisors, Prof Georgios Theodoropoulos and Dr Rami Bahsoon, for their unwavering support in the course of my PhD studies. Their guidance, enthusiasm, and encouragement have been invaluable to me. The fruitful discussions with them have always been a source of inspiration, which guides me to the quest for research challenges at fundamental levels.

I would like to thank Dr Nikos Tziritas, for the suggestions and discussion in my research work. I would also like to thank the thesis group members, Prof Peter Tino and Dr Leandro Minku, for their constructive questions and suggestions for the research work and progress.

It is also my great fortune to work with wonderful people both at Birmingham and SUSTech. I would like to thank Hargyo T. N. Ignatius for all the insightful discussions, both online and offline. Special thanks to the team in SUSTech: Zhengchang Hua, Jingran Shen, Shuyi Chen, Georgios Diamantopoulos, and Christian Vergara. The group meeting has always been an extraordinary opportunity to communicate and exchange wonderful thoughts. Also, many thanks to Yaru Zhao for the support of all the tedious administrative tasks.

I would like to thank my parents for their everlasting care and unconditioned support. And lastly, special thanks to my fiancée Jiayu Wang, who has always been a great source of support throughout the course of my PhD studies.

Contents

List of Figures

List of Tables

Acronyms

A/IS Autonomous/Intelligent System. 25

AI Artificial Intelligence. 3, 57, 58, 60, 67, 162, 172

AMR Autonomous Mobile Robot. 131, 141, 144, 145, 147–151, 153–155, 165

CDT Cognitive Digital Twin. 18, 20, 56, 59, 66–68, 81

CPS Cyber-Physical Systems. 8, 27, 41, 56, 131–135, 139

DDDAS Dynamic Data-Driven Applications Systems. 14, 18, 21, 35–37, 39, 79, 80, 84, 87, 135, 143, 160–163, 165, 166, 168, 176, 181

DT Digital Twin.

GRL Goal-oriented Requirements Language. 139, 145, 146, 149, 151, 159

IIOT Industrial Internet of Things. 57, 58

IoT Internet of Things. 8, 11, 32, 41, 54, 57, 86, 131, 141, 143, 163

MAPE-K Monitor-Analyse-Plan-Execute over a shared Knowledge. 26–28, 41, 60, 139

MAS Multi-Agent Systems. 6, 7, 13, 32, 50, 51, 90, 128, 143

ML Machine Learning. 37, 57, 58, 61, 63–66, 78, 79, 166–168, 170, 177

P2V Physical-to-Virtual. 14–20, 23, 32, 39, 81, 130, 174, 179, 182, 183

UAV Unmanned Aerial Vehicle. 65, 79, 88, 163

V2P Virtual-to-Physical. 14–19, 21, 23, 32, 39, 130, 174, 179, 180, 182, 183

Chapter One

Introduction

The concept of Digital Twin (DT) has received increasing attention in recent years and has been explored in various fields, including smart manufacturing, logistics, transportation, city and urban planning, smart agriculture, etc. [117, 173, 23, 89]. A DT is generally regarded as the virtual representation of existing physical assets, products, objects, or systems [173, 233]. The digital twin as a virtual representation can further benefit the physical twin with analysis and services supported by real-time simulation. Many definitions of the DT have been proposed in the literature, each firmly based on application context [23, 233]. A representative and inclusive definition of the DT, which is adopted in this thesis, has been put forth in [14]:

> *"A digital twin is the combination of a computational model and a real-world system, designed to monitor, control and optimise its functionality. Through data and feedback, both simulated and real, a digital twin can develop capacities for autonomy and to learn from and reason about its environment."*

Despite the various definitions, this thesis views the Digital Twin paradigm, in general, as a system consisting of the *physical world*, *models* of the physical world, and the real-time

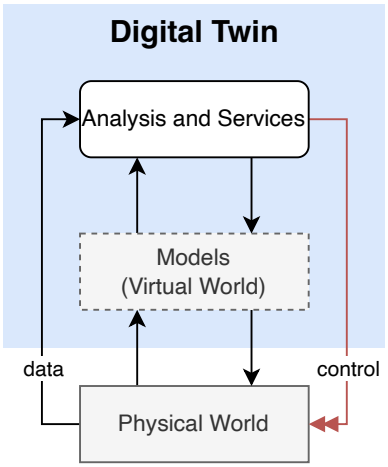


Figure 1.1: The Digital Twin Paradigm.

analysis capability enabled by those models (See Figure. 1.1). The physical world is closely entangled with its models through real-time data exchange. The physical world may contain software systems and the physical environment. For instance, for a smart city digital twin, autonomous cars can be the software-based system operating in the physical world, while road traffic is the physical environment. The digital twin can provide simulation analysis and a holistic view of the entire city to optimise traffic and the behaviours of autonomous cars.

The focus of analyses and services enabled by DTs has been geared towards simulation and what-if analysis to inform deployment and subsequent refinements of the physical systems through continuous monitoring and data assimilation [63, 237, 37]. An essential added value of the DT for the real-world system is its capability for more informed runtime decision-making and planning through simulation [214, 78]. Specifically, the added value for the real-world system can be summarised according to [196]:

> "*Digital twin can be defined as a virtual representation of a physical asset enabled through data and simulators for real-time prediction, optimization, monitoring, controlling, and improved decision making.*"

These capabilities of real-time analyses and services imply the idea of an *intelligent DT*. A DT can be designed to incorporate artificial intelligence (AI) to realise an autonomous system, thus enabling services such as optimisation and automatic control of the real-world system [15, 251]. An intelligent DT should also incorporate "self-*" capabilities such as self-optimising or self-configuring, thus making decisions autonomously and involving less human intervention [173]. It can be argued that "self-*" (self-adapting, self-regulating, self-monitoring, self-diagnosing, etc.) should be a distinguishing feature that characterises the intelligence of the DT [171].

Despite the efforts in characterising and designing intelligent DTs, most of the literature only focuses on the intelligence on the digital part: how the DT can support more informed analysis, learning, and services in an autonomous fashion. When integrating intelligence and DTs, it is common to regard intelligence as solely manifested in the digital part of the DT-based system. However, inadequate attention has been paid to the intelligence of the real-world system in the Digital Twin paradigm. This calls for in-depth analyses of the synergy between intelligence and Digital Twin, and how intelligence can be instantiated in different components in the DT-based system.

1.1 Intelligence and Digital Twins

This thesis regards intelligence and DTs as two equally important concepts. This section discusses the possible fusion of intelligence and DTs. It can be argued that "intelligence" in the DT realm is not a characteristic that can only be embodied in the digital part, but also the real-world system.

A classification for the different combinations of intelligence in the digital part and the real-world system is provided in Figure 1.2. In the figure, the level of intelligence is

classified into a dichotomy of "intelligent" and "naive". The real-world system and its DT can be either intelligent or naive (as shown in the horizontal and vertical axes of Figure 1.2):

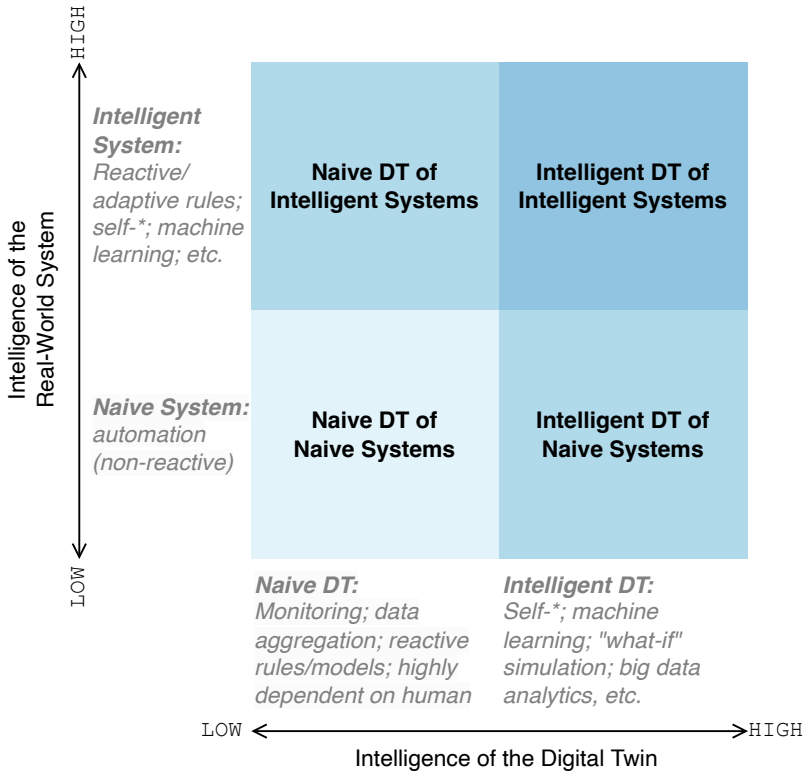


Figure 1.2: A matrix delineating the intelligence of the DT and the real-world system.

- A *naive DT* represents a DT with little intelligence and autonomy. This means its main purpose is to monitor/estimate the state of the physical twin or act as a platform for real-time data aggregation. For a naive DT, its functioning and decision-making for the real-world system highly rely on human intervention. It may use pre-built rules or simple decision models to formulate decisions, but does not support predictive analysis and real-time simulation evaluation.

- An *intelligent DT* means it is autonomous, self-* and can accumulate knowledge through learning. It can predict future states of the physical world and can support real-time what-if analysis based on simulation models of the physical world. More im-

portantly, a highly intelligent DT can not only optimise the real-world system through analyses and services but also self-adapt its own models and analysis techniques.

- A *naive system* refers to a real-world system that has the minimum level of computation and intelligence or highly relies on human control. This type of system is not able to perceive its environment nor respond accordingly. One typical example is the automated robotic arms in an assembly line. These robotic arms can only follow pre-programmed procedures but cannot react to surrounding events. Robots whose behaviours highly rely on manual control from humans, e.g. a robot whose movement can only be controlled through teleoperation from human users, are also regarded as naive systems.

- An *intelligent system* refers to a real-world system that can perceive its environment and act accordingly with computation analysis, or even support learning and data analytics. It can act self-adaptively by changing its behaviour based on its surroundings in a responsive manner. One example is autonomous cars, whose onboard control can enable autonomous manoeuvre according to road scenarios in real-time. Autonomous cars may also accumulate knowledge of driving after it has experienced various road scenarios.

Therefore, according to the above dichotomy for the intelligence of both the DT and the real-world system, marrying intelligence with DTs will result in the following four fundamental categories (as shown in Figure 1.2):

- *Naive DTs of naive systems.* Some typical examples can be structural health monitoring for aircraft wings [144] or visualisation of a manufacturing plant [276]. The intelligence of both the DT and the system is limited.

- *Intelligent DTs of naive systems.* This category is commonly discussed in smart manu-

facturing and many other domains. To give an example, a DT can utilise both simulation data and real-time data, as well as predictive analysis, to facilitate the prognostic and health management (PHM) of a manufacturing system [231]. Here, the intelligence of the manufacturing system is limited, while the DT can support more sophisticated analysis to optimise the functioning of the manufacturing system. Another example is in [156], which studies DTs for robots. Robots are controlled by teleoperation issued by human users, thus being naive. The DT can utilise simulation to predict the behaviour of robots, and it can monitor and issue commands (emergency stops) to the physical twin (the robots).

- *Naive DTs of intelligent systems.* This type can be counter-intuitive, but for an intelligent real-world system, its DT can be naive due to the necessity of human intervention. For instance, an autonomous taxi may encounter corner cases or emergency events, such as traffic accidents ahead or road maintenance. These scenarios can sometimes be difficult for autonomous taxis to manage gracefully due to the lack of training data during the offline development phase or the incapability of onboard policies for driving behaviours. A DT can monitor these unexpected scenarios, and hand over the control of the vehicle to a remote human driver sitting in a remote centre, via low-latency communication such as 5G.

- *Intelligent DTs of intelligent systems.* This category enables the DT to assist the intelligence of the real-world system autonomously with more informed what-if simulation analysis and learning capabilities that cannot be supported onboard the real-world system due to issues such as energy consumption or resource constraints. The DT can also self-adapt its own analysis models.

Similar to the above discussion, there has been preliminary work that investigated the relationship and synergy between agents, multi-agent systems (MAS) and digital twins

[163]. In [163], the authors propose two views of the synergy: how MAS can utilise digital twins, and how digital twins can utilise MAS. In the first view, MAS can utilise DTs as a virtual environment or platform, in which DTs act as a shared medium for agents to access physical assets and resources. In the second view, DTs can utilise agents to offer more intelligent services and functionalities. However, their perspectives generally fall into the category of *intelligent DTs of naive systems*. The real-world system is naive because it only involves physical assets such as patients, rescuers and ambulances. The other three categories discussed in this section are not involved in [163].

Among the four combinations of intelligence and DTs discussed above, this thesis specifically focuses on the vision of ***intelligent DTs of intelligent systems***. Such a paradigm is promising since it can exploit both the computational capability of the real-world system and the DT, thus reaching much higher levels of autonomy and possibly enabling human-like cognitive behaviours. However, this combination has received little attention in the literature.

1.2 An Illustrative Example

To investigate the vision of *intelligent DTs of intelligent systems*, it is essential to clarify the reasons behind the necessity of this vision, highlight its advantages, and identify any outstanding issues that need to be addressed. This section peeks into an example in Cyber-Physical Systems (CPS) and illustrates the advantage of DT, and how intelligence can be embodied in both the real-world system and the DT.

As computing infrastructure becomes less expensive and more energy-efficient, and digitalisation becomes more prevalent, smart computing nodes (e.g. edge servers, autonomous robots) have been increasingly adopted in various domains such as the Internet of Things

(IoT), edge computing, and ubiquitous computing. These computing nodes can be made intelligent and autonomous to finish tasks without human intervention. They self-adaptively change their behaviour and can even acquire new knowledge in the course of operation. One example can be autonomous drones for decentralised collaborative surveillance in the city. Drones can use their onboard intelligence to collaboratively identify which area is of interest (e.g. high density of crowd, traffic congestion) and negotiate with other drones to decide who will monitor which area. Drones can also gain knowledge about the behaviour patterns of crowds or traffic by analysing historical observations, which can be used to predict potential areas of interest in the future.

These systems are essentially complex smart Cyber-Physical Systems (CPS), which are characterised by the close entanglement between software and physical hardware. In CPS, information from the physical world is continuously collected, and *computation*, *communication*, and *control* (3C) enabled by software (the "cyber" part) manage the behaviour of the physical systems (the "physical" part) in return to perform dedicated tasks autonomously and collaboratively [195].

Autonomy is an important dimension in complex CPS that enables the system to support intelligent analysis and perform specific tasks in dynamic and complex environments without human intervention [174, 39, 140]. Knowledge management is an essential capability to support autonomous behaviours [101, 26]. The software part should be able to enable **self-awareness**, thus acquiring and updating knowledge continuously and utilising the knowledge learned to change the behaviour of the system self-adaptively [101, 26]. Intelligence can be distributed to each subsystem/node of the entire CPS, where each subsystem (node) is autonomous and intelligent on its own and makes decisions by itself [75]. The software of each node collects data from the physical part of the node, environment and the software part itself, to formulate autonomous adaptation on the behaviour of the CPS.

These autonomous CPS may work in open and dynamic environments, facing unpredictable events and sometimes involving interaction with humans. In the drone surveillance example described earlier in this section, the city may involve a large number of entities, such as humans and cars. There can also be unpredictable emergency events like car accidents, which increases the difficulty for drones to manage. However, subsystems/nodes (e.g. autonomous drones) of the CPS may have limited hardware resources (e.g. CPU and memory) or energy consumption constraints (if they are battery-powered), which can limit their ability to perform intensive computational analysis [26]. As a result, the nodes may only support a limited level of intelligence. Such constraints can significantly impede the capacity of each node to make optimal decisions in real time to mitigate runtime uncertainties in the environment. With constrained intelligence, the following challenges may arise:

- **Trade-off among different goals.** CPS are designed to perform specific tasks in the physical environment. In addition to the task, the system may also be required to achieve various other quality goals, business goals or regulatory compliance, such as safety, efficiency, profit, carbon emission, etc. These goals can usually be conflicting, thus requiring trade-off analyses. For instance, an autonomous car in the city may need to balance the goals of reaching the destination fast while ensuring safe driving. Such trade-off needs to be analysed online to react to different scenarios, which may be beyond the capabilities of intelligence onboard, or the nodes' limited computational resources may not support generating timely solutions.

- **Dynamic and open environment.** For highly dynamic and complex environments, it can be impossible at design time to foresee or enumerate all possible environmental contexts and design the control scheme accordingly. The designed intelligent control scheme of each node may only be applicable in specific scenarios, since not all possible future threats can be foreseen during the design phase, and the full behaviour of the complex environment only emerges during operation [213, 238]. The existence of

human and machine-related errors and their ripple impact on the system can best be understood at run-time due to the absence of necessary contextual information during design time. Also, the best trade-off between multiple goals may be different in different environmental contexts and states.

- **Dynamism of the CPS.** CPS are usually systems of systems, and the interaction between multiple subsystems increases the complexity of the overall system behaviour. The CPS itself may involve a network of heterogeneous nodes that differ in their computational capabilities and constraints. The network may change during the task operation, with nodes joining and leaving the network dynamically. This requires a proper design of the self-adaption logic that is aware of the intrinsic dynamism of the system.

The composite effect of the above challenges makes it difficult for resource-constrained CPS to handle the uncertainties only known at runtime. Although cloud-based solutions have been proposed for resource-constrained CPS to offload the computation originally onboard the CPS to the cloud [154, 273, 104, 40], the complexity of the environment at runtime calls for high-fidelity *simulation* to *accurately* predict and understand outcomes of candidate decisions in real-time for more in-context evaluation.

The Digital Twin paradigm can be a promising approach for these constrained intelligent systems. A Digital Twin can model and replicate the entire physical space in real time. With the support of cloud computing or high-performance computing, the DT can utilise abundant computational resources to provide holistic analysis by simulating the entire environment, offering more informed decision-making for the resource-constrained smart computing nodes. In comparison, it can be difficult for resource-constrained computing nodes themselves to conduct complex simulation analysis. Also, the Digital Twin can augment the functionality of the pre-existing system with minimal design modifications of that system,

which reduces the effort in re-developing the behavioural logic of the computing nodes.

The Digital Twin should also be made intelligent and autonomous. In open and highly dynamic application environments such as city surveillance, decisions may need to be made in a timely manner. Such a requirement for fast decision-making in a complex environment may be difficult for humans to manage and analyse. Therefore, the DT should be made self-adaptive and autonomous to make decisions by itself to optimise the real-world system. This leads to the need for engineering *intelligent DTs of intelligent systems*.

There has been only a few preliminary work that focuses on intelligent DTs of intelligent systems. For instance, Sun, Lei, Wang, Liu, and Zhang [225] use DTs to optimise the training efficiency of federated learning (FL) on IoT devices. They use offline-trained deep reinforcement learning to dynamically change the aggregation frequency of FL based on the current training state of FL and the status of the devices. Zhang, Cao, and Zhang [270] focus on using DTs to facilitate multi-agent learning in vehicular edge computing. The DT uses a gravity-model-based scheme to aggregate agents into multiple groups to improve the learning efficiency of agents. The multi-agent learning results, in return, adjust the aggregation scheme of the DT. Despite these existing efforts in the exploration of *intelligent DTs of intelligent systems*, DTs in the existing work do not involve simulation what-if analysis, which is an essential capability addressed by the DT to support more sophisticated analysis [93].

Research is still inadequate in applying DTs that use *online what-if analysis with high-fidelity simulation* for *self-** intelligent systems. Such a combination poses new requirements firstly for the digital twinning of the system: the intelligence/knowledge management processes of the real-world system should also be replicated by the DT. This is essential to ensure the simulation prediction of the entire system is valid, since the system's behaviours are highly dependent on its intelligence and runtime knowledge. This requirement is not

needed for the work [270, 225] since no simulation is involved in their DTs. New mechanisms are also needed to coordinate the intelligence of the DT and the intelligence of the system to enable ***mutual intelligence enrichment***: the intelligence of the DT assists and perfects the intelligence of the real-world system, and vice-versa.

1.3 Aim and Research Questions

As discussed in the previous section, the problem of existing research lies in the inadequate investigation and understanding of *intelligent DTs of intelligent systems*. Therefore, the aim of this thesis is to formulate the novel notion of *intelligent DTs of intelligent systems*, characterise the symbiotic relationship of mutual intelligence enrichment, and identify the challenges and solutions. Specifically, this thesis aspires to design a framework for the coordination of knowledge and the management of knowledge between the digital twin and the physical twin, such that the intelligence of both twins can be mutually enriched. The benefit of mutual intelligence enrichment is that the DT can utilise real-time simulation analysis to acquire new knowledge and subsequently instruct more informed adaptation of the real-world system, while the real-world system's data and the real-world system's own knowledge perfect the intelligence of the DT.

1.3.1 Scope

This subsection specifies the thesis's perspective on DT and the scope for intelligence in real-world systems. The DT can augment the intelligence of the physical system with more informed analysis in real-time, while the physical system can provide a more heightened level of realism with sensor data to enrich the knowledge of the DT, forming a real-time feedback loop that continuously perfects both digital and physical worlds. Throughout the thesis, we

will use the terms "real-world system" and "physical system" interchangeably to refer to the software-based system in the physical world.

This thesis refers to intelligent systems in the physical world as autonomous complex CPS whose software part is composed of intelligent agents, forming a multi-agent system [259]. The intelligence of each agent is characterised by "self-*" (self-configuring, self-healing, self-optimising, self-protecting, etc.) adaptive capabilities [107, 212]. This thesis specifically focuses on one type of self-*: **self-awareness**, which can be regarded as an enabler of self-adaptation and other self-* properties [212]. Self-awareness refers to the system's ability to acquire and utilise knowledge dynamically [135]. A more detailed description of self-awareness will be given in Chapter 2. Therefore, an intelligent system would be illustrated as a large-scale autonomous system consisting of connected computing nodes (e.g. drones), each of which is managed by an onboard intelligent agent. Each node can accumulate knowledge about itself and the environment to make its own decisions and collaborate with other nodes in the network. The computation within the entire system is distributed, and the control is decentralised to each agent [65, 42, 257]. The degree of autonomy and level of intelligence of these agents may be limited by issues like energy consumption and the computation capability of each node.

In this thesis, a Digital Twin is regarded as a simulation model of a real-world system and its physical environment, combined with the services supported by the model. The DT has a symbiotic relationship with the physical world: its model is continuously updated by real-time data sensed from the physical world, and the DT can use online simulation analysis to provide new insights for human managers/users and also support autonomous control of the system in real-time. The runtime/online analysis is enabled by simulating different what-if scenarios on the possible assumptions, inputs, environmental conditions, etc., to predict their future outcome. By "online" or "runtime", this thesis refers to the ability to conduct analysis and obtain insights in a timely manner while the system is running and the

environment is evolving. This is in contrast with "offline" or "design time", which refers to the activity before a system is deployed in the real environment. With sensor data from the physical system, the DT empowers intelligence with more informed analysis to assist and help the real-world system that has constrained intelligence.

The symbiotic relationship between the DT and the physical world is enabled by real-time feedback loops. A DT can utilise principles of Dynamic Data-Driven Applications Systems (DDDAS), which incorporate data into an executing model to improve its accuracy, and the model, in return, guides the measurement process [60]. In the feedback loop, the DT assimilates real-time data to the model, updates the model to ensure equivalence, acquires new knowledge with the up-to-date model and sensor readings, and informs new insights or adaptations of the real-world system. The feedback loop can be characterised by two directions of data flow: Physical-to-Virtual (P2V) and Virtual-to-Physical (V2P) [233]. P2V describes the data flow and associated mechanisms from the physical world to the DT, while V2P refers to the direction of data flow from the DT to the physical world.

Notice that the term "knowledge" referred to in this thesis, which enables the intelligence of the DT and the real-world system, does not pertain to *domain knowledge*. Instead, this thesis specifically focuses on *runtime knowledge* that is dynamically acquired through the DT's or real-world system's own perception from sensors. The sensors can be either internal (obtaining data about itself) or external (obtaining data about its environment) [143]. Such kind of runtime knowledge is closely related to the self-awareness capability of the DT or the real-world system: how the DT or the real-world system understand itself and its environment.

One can argue that it is not new to use the DT's what-if analysis enabled by simulation models to optimise a physical system in real-time [214]. However, research is still inadequate in enabling the DT to use real-time simulation analysis to assist a real-world system that can

self-adapt its behaviour and may learn knowledge on its own. In this way, the DT does not need to have full control of the real-world system, thus reducing the computation burden of the DT and also the communication overhead. It remains to be investigated how DTs can play a role in this context, and how the intelligence of the DT and the intelligence of the physical system can coordinate.

1.3.2 Research Questions

Further research is needed to investigate the novel concept of mutual intelligence enrichment. An intelligent DT is expected to overcome the computation limitation of a multi-agent self-aware system, thus empowering intelligence. Specifically, the following research questions are to be answered:

- **RQ1** How to design an intelligent DT that combines knowledge management and simulation what-if analysis to assist self-aware systems?
- **RQ2** How to achieve mutual intelligence enrichment between the intelligent DT and the self-aware systems?
- **RQ3** If humans are involved in the mutual intelligence enrichment process, how can DTs provide explainability to humans for improved decision-making?

1.4 Objectives

One fundamental aspect of the DT paradigm is the real-time feedback loop, which is composed of V2P synchronisation, P2V control and the intelligence that informs the above two processes. Driven by the research questions, the following objectives will be addressed:

- **O1**. Develop a generic reference architecture for the vision of *intelligent DTs of intelligent systems*. This reference architecture will identify the essential modules and components and serve as the basis for the subsequent solutions to other research questions.

- **O2**. Refine the generic reference architecture to reflect (1) the synergy between fine-grained knowledge management and real-time simulation what-if analysis of the DT and (2) explainability-driven decision-making for human-in-the-loop. This objective addresses **RQ1** and **RQ3**.

- **O3**. Develop consolidated mechanisms for the components and processes of the real-time feedback loop in the generic reference architecture. The solution to mutual intelligence enrichment is divided into two dimensions according to the directions of data flow between the DT and the physical world:

  - **O3.1** Physical-to-Virtual (P2V): the novel mechanism should enable the DT to utilise knowledge managed by the real-world system to maintain equivalence of the model adaptively and efficiently.
  - **O3.2** Virtual-to-Physical (V2P): the DT should be able to utilise simulation and its own knowledge to provide insights that cannot be learned by the self-aware system in the real world alone, and suggest more informed adaptations of the system.

  Such a division focuses on the two directions of mutual intelligence enrichment separately. Mechanisms designed for each direction should improve the adaptivity or knowledge management capability of the physical/digital twin. This objective addresses **RQ2**.

- **O4**. Instantiate and evaluate the proposed mechanisms in **O3** with experimental evidence and analysis. This objective addresses **RQ1** and **RQ2**.

1.5 Research Methodology

This thesis follows a similar research methodology to the design science methodology for information systems research as suggested in [191]. The methodology of this thesis is divided into five iterative steps:

1. **Problem identification and motivation:** The motivation lies in utilising DTs to optimise the behaviour of a real-world system with a certain level of intelligent capabilities. This thesis conducted a literature review of how intelligence is involved in DTs-related research from two perspectives: the intelligent capabilities of a DT in different dimensions, and how intelligence has been addressed in the physical twin. Such review is presented later in Chapter 4 and Chapter 5, respectively. This literature review has identified that the research effort is limited in addressing the scenario where intelligence exists in both the DT and the real-world system. The general problem is, that the role and capabilities of DT on how it interacts with the intelligent system are not clearly defined. One of the key dimensions of intelligence is knowledge. Self-aware systems offer a systematic approach to architect knowledge-related decision-making behaviours at different fine-grained levels, which serves as the basis for defining an intelligent system in this thesis. Therefore, given the nature of DT requires two-way interaction with the real system, the more specific problem addressed by this thesis is that it is unclear how the knowledge of the real system and the DT can be mutually utilised and enriched by each other.

2. **Objectives and hypothesis for the solution:** Motivated by the problem, reference architectures and mechanisms are needed to delineate the role of DT and the interaction between the intelligent DT and the self-aware real system. Specifically considering the two-way interaction, the solution should enable mutual intelligence enrichment from the V2P and P2V perspectives, and also consider the factor of human-in-the-loop.

Mechanisms involved in the P2V aspect should enable knowledge maintained by the real-world system to facilitate cost-efficient and adaptive equivalence maintenance for the models of the DT. The V2P mechanism should be able to leverage knowledge gained through simulation analysis by the DT to perfect the decision-making of the self-aware systems. Based on the objectives of the perspectives mentioned above, the hypothesis is made for the possible solutions: in P2V, the designed mechanisms are expected to improve the continuous update process with respect to quality properties; in V2P, the designed mechanisms are expected to reveal new insights into the optimisation of the real-world system.

3. **Design and development:** The process of design and development is oriented toward refining from a generic design solution to perspective-specific solutions with more detailed configurations. This thesis first designs reference models and architectures to delineate a generic framework of the DT-based system that addresses intelligence in both the physical twin and digital twin. Then, the components and processes involved in the generic design are refined to reflect the following specific aspects respectively: the intelligence of the DT, the P2V process, the V2P process and the human-in-the-loop factor. To integrate runtime knowledge management and simulation analysis to the intelligence of the DT, expertise from self-aware computing systems and DDDAS is utilised to propose a reference model towards engineering a cognitive DT. Therefore, not only the real-world system is self-aware, but the DT is also self-aware. Then the generic reference architecture is refined with detailed modules, mechanisms and algorithms for P2V and V2P from the runtime knowledge perspective.

4. **Demonstration:** The designed reference architecture and mechanisms are instantiated using multiple case studies. For the design of the self-aware DT, we justify the applicability of how it can be mapped to a smart city drone-based logistics example. For P2V equivalence, the refined architecture and algorithms are instantiated through

a demonstrative example of smart mobile cameras for distributed surveillance. The DT-based system is implemented using two simulators. One simulator represents the physical world, and another serves as the DT. For V2P, a smart warehouse logistics scenario is used to demonstrate the necessity of the DT-enabled runtime intelligence enrichment for self-aware systems. The applicability of human-in-the-loop is also justified through the smart warehouse logistics scenario.

5. **Evaluation and validation:** According to [19], validation refers to "substantiating that the model, within its domain of applicability, behaves with satisfactory accuracy consistent with the study objectives". This thesis utilises simulation-based experiments to validate the proposed models (mechanisms) of the DT against the elicited objectives and hypothesis. Specifically, the effectiveness of the proposed P2V and V2P mechanisms are quantitatively evaluated through controlled experiments conducted with simulation tools.

1.6 Thesis Contributions

This thesis has developed novel reference architectures and mechanisms to realise the mutual intelligence enrichment for *intelligent DTs of intelligent systems*. A generic reference architecture is proposed to illustrate the essential components and interaction between the physical and digital twins. This provides an overall solution to the design of intelligent DTs of intelligent systems. Then, the generic reference architecture is refined accordingly to reflect the specific requirements of each research question and objectives. This thesis designs mechanisms and functionalities of the modules in the refined architectures to solve the challenges related to each research question. Finally, these solutions are evaluated respectively with simulation analysis.

Specifically, the following gives an overview of the contributions of the thesis:

- **Clarifying the synergy between DTs and intelligence.** This thesis explores the potential of infusing intelligence into both the DT and its real-world counterpart system. A four-category classification is proposed to demonstrate how the intelligence of the DT and that of the real-world system can combine.

- **Digitally twinning intelligent self-aware systems.** This thesis formally describes how a multi-agent self-aware system can be modelled and replicated within the Digital Twin paradigm. Especially, the modelling scheme provides a novel perspective on replicating the knowledge and the knowledge management processes of the system to the digital twin.

- **A generic reference architecture for intelligent DTs of intelligent systems.** Based on the digital twinning scheme, this thesis proposes a generic reference architecture that depicts the essential components within the DT and their interactions. This contribution addresses **RQ1** and **O1**.

- **Engineering intelligent DTs with self-awareness**. This thesis proposes refinement to the generic reference architecture and maps self-awareness design principles to the intelligence of the digital twin. The mapping leads to a reference model for a cognitive DT that combines dynamic fine-grained knowledge management, simulation prescriptive what-if analysis, and autonomy. This contribution addresses **RQ1** and **O2**.

- **P2V: knowledge equivalence and knowledge equivalence checking**. This thesis proposes the novel notion of knowledge equivalence for digitally twinning knowledge-aware systems. A cost-efficient approach is proposed to identify the discrepancy of the model online with knowledge equivalence checking. In this way, the knowledge

of the real-world system facilitates the DT for more efficient model updates. This contribution addresses **RQ2** and **O3.1, O4**.

- **V2P: goal-aware DT-enabled knowledge enrichment for rule-based systems**. By leveraging the proposed self-aware DT reference model, this thesis designs a goal-aware DT approach to adapt rule-based self-aware systems for runtime trade-offs in the context of regulatory compliance. This approach enables the intelligent DT to enhance the intelligence of the real-world system. The benefits of this approach are evaluated with a smart warehouse logistics example. This contribution addresses **RQ2** and **O3.2, O4**.

- **A reference architecture supporting explainability and human-in-the-loop**. This thesis proposes an explainability-driven reference architecture for human-in-the-loop decision-making of the digital twin. The reference architecture is based on the generic reference architecture and combines principles of DDDAS. This contribution addresses **RQ3** and **O2**.

1.7 Publications

The research contributions and results of this thesis have been published in several peer-reviewed research papers, as detailed below. This thesis should be the definitive account of the work presented in the following publications.

- N. Zhang, R. Bahsoon, and G. Theodoropoulos, 'Towards Engineering Cognitive Digital Twins with Self-Awareness', in 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Oct. 2020, pp. 3891–3896.

  Part of Chapter 4 is derived from this publication. In particular, Sections 4.2.3, 4.3-4.7

should be the definitive account of this publication.

- N. Zhang, R. Bahsoon, N. Tziritas, and G. Theodoropoulos, 'Explainable Human-in-the-loop Dynamic Data-Driven Digital Twins', in 4th International Conference on InfoSymbiotics/Dynamic Data Driven Applications Systems (DDDAS2022), Nov. 2022, pp. 233–243.

  Chapter 7 is based on this publication.

- N. Zhang, R. Bahsoon, N. Tziritas, and G. Theodoropoulos, 'Knowledge Equivalence in Digital Twins of Intelligent Systems', ACM Transactions on Modeling and Computer Simulation (TOMACS), vol. 34, no. 1, pp. 3:1-3:37, Jan. 2024.

  Chapters 3 and 5 are derived from this publication. Part of Chapter 2 is also based on this publication, in particular Sections 2.1 and 2.2.

- N. Zhang, R. Bahsoon, N. Tziritas, and G. Theodoropoulos, 'A Digital Twin Approach for Adaptive Compliance in Cyber-Physical Systems: Case of Smart Warehouse Logistics', IEEE Transactions on Systems, Man, and Cybernetics: Systems, (Sep. 2023), **(Under Review)**.

  Chapter 6 is based on this publication.

1.8 Structure of the Thesis

The section presents the thesis structure. In particular, Chapter 2 provides a generic background and related work, while each subsequent chapter includes related work with the specific topic of that chapter.

- **Chapter 2** provides a generic background and related work of this thesis, including

the essential background on digital twins and self-aware computing. Similar concepts related to DTs are also explored.

- **Chapter 3** formally models the physical twin, which is a multi-agent self-aware system, and describes how to build a digital twin of such a system. A generic reference architecture for intelligent digital twins of intelligent systems is then proposed, which serves as the foundation of this thesis' contribution. The following four chapters will provide further refinement to the reference architecture by focusing on its modules and processes. This chapter addresses **O1**.

- **Chapter 4** specifically focuses on engineering the intelligence of the digital twin. This chapter first classifies the intelligence of the DT in the existing literature by their capabilities. Then, self-awareness design principles are applied to the analysis and decision-making-related capabilities addressed by the generic reference architecture in Chapter 3. Such an effort leads to a refined reference architecture that focuses on the coordination of dynamic fine-grained knowledge management and online simulation what-if analysis. This chapter addresses **O2** and **RQ1**.

- **Chapter 5** focuses on the P2V data flow of the real-time feedback loop in the generic reference architecture in Chapter 3. It discusses the equivalence-related issues between the model and the self-aware system, especially the novel notion of knowledge equivalence. This chapter then proposes how to check whether the model is equivalent to the real world cost-efficiently based on the method of knowledge equivalence checking. A smart mobile camera application scenario is used to evaluate the efficiency of Knowledge equivalence checking. This chapter addresses **O3.1**, **O4** and **RQ2**.

- **Chapter 6** focuses on the V2P data flow of the real-time feedback loop in the generic reference architecture. Also leveraging the self-awareness capabilities proposed in Chapter 4, this chapter provides the mechanism for DT-enabled goal-aware adaptation

to the rules of the real-world system. The necessity of such a DT approach is evaluated in the adaptive compliance management in smart warehouse logistics. This chapter addresses **O3.2**, **O4**, **RQ1** and **RQ2**.

- **Chapter 7** considers the scenario where humans are involved in the decision-making of the digital twin. The explainability of the decisions generated by the digital twin is needed to improve the trust of humans and facilitate human intervention. This chapter proposes an enriched design based on the generic reference architecture by identifying where explanations are needed in the real-time feedback loops of the digital twin. This chapter addresses **O2** and **RQ3**.

- **Chapter 8** concludes the thesis and discusses the findings with open issues.

Chapter Two

Background and Related Work

This thesis specifically considers that the real-world system possesses one type of intelligence capability: self-awareness. This chapter first provides a background starting from intelligent systems and describes in detail what a self-aware system is to set the foundation of research in later chapters. Then, an overall background is provided for the Digital Twin paradigm and other similar concepts. This chapter mainly provides the general background and related work for intelligent DT and self-aware systems. More detailed related work on the different dimensions of the research is provided individually in each of the later chapters.

2.1 Intelligent Systems

This chapter starts with generic definitions of the intelligent system, and describes the architecting effort of intelligent systems towards self-*, especially self-awareness capabilities.

According to IEEE Standard 7010-2020, An *autonomous/intelligent system (A/IS)* is defined as "*a semi-autonomous or autonomous computer-controlled system programmed to carry out some task with or without limited human intervention capable of decision making by independent inference and successfully adapting to its context. An example is an A/IS*

that refers to a computer system instantiated in a product or service" [111]. This definition correlates autonomy to the intelligence of systems. Further, an *autonomous system* defined by IEEE standard 7001-2021 refers to "*a system that has the capacity to make decisions itself in response to some input data or stimulus with a varying degree of human oversight or intervention depending on the system's level of autonomy*" [110]. Therefore, the emphasis of intelligence is the system's capability to independently plan for its behaviour in a dynamic environment.

2.1.1 Self-Adaptive Systems with MAPE-K

In designing intelligent capabilities, IBM proposed the concept of *autonomic computing* in 2001, which compares complex computing systems to the human autonomic nervous systems and promotes self-governing properties to be incorporated into the system design [107, 109]. The vision was put forth under the increase in the number of interconnected computing devices that operate cooperatively. It is infeasible for system administrators to anticipate the behaviour of interactions, thus requiring runtime adaptive capabilities for the configuration, optimisation and maintenance of such complex systems [128]. An autonomic computing system should have self-management capabilities to change in accordance with business policies and objectives, such as: self-configuration, self-optimisation, self-healing, and self-protection [108]. Each system component is architected with a control loop denoted as MAPE-K with a total of five components: Monitor (M), Analyse (A), Plan (P), Execute (E), and Knowledge (K) [108]. The M, A, P, E components form a loop that identifies the data and control flow for the process of reacting to events. The four components all interact with the K component, and knowledge acts as the basis of reasoning and decision-making.

The manageability problem under complexity that autonomic computing aims to tackle motivates the study of self-adaptive systems [255]. A self-adaptive system uses closed-

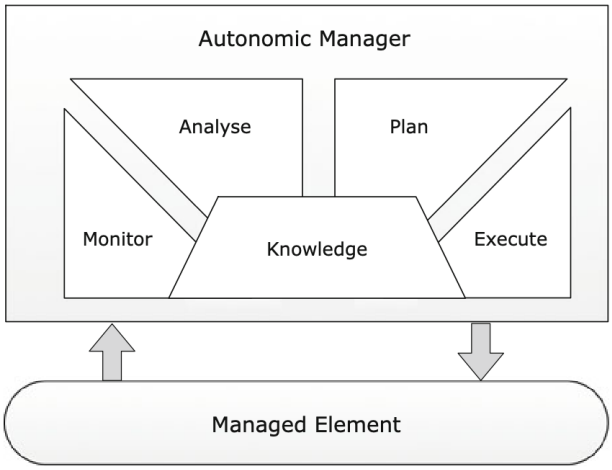


Figure 2.1: The MAPE-K structure of the autonomic manager, which is the primary building block of autonomic systems (Figure from [256]).

loop control to autonomously adjust its behaviour or structure in response to the dynamic environment and the awareness of itself [51, 141]. *Self-adaptation* as an intelligent "self-*" capability is also highly influenced by the MAPE-K architecture. The self-adaptive design has been applied in various domains, such as cloud computing [48], cyber-physical systems [174], etc.

A variety of control schemes are possible when combining multiple self-adaptive entities. The MAPE-K control components can also be decentralised. Different software entities can have all or part of the full MAPE-K components and interact with each other. Different decentralised control patterns based on MAPE-K have been investigated in [257]: coordinated control pattern, information sharing pattern, master/slave pattern, regional planner pattern, and hierarchical control pattern.

2.1.2 Self-Aware Computing

The knowledge repository in the MAPE-K loop and self-adaptation, in general, is an important component for decision-making and is central to intelligent systems. The maintenance of such a knowledge repository is specifically addressed by *self-aware computing* [143]. ***Self-awareness*** can be regarded as the enabler of self-adaptation and the primitive level among all self-* properties, meaning the system is aware of the state of itself and environment contexts, in order to extract, accumulate and maintain knowledge through interaction with the environment and other computing devices [71, 212].

In [135], self-aware computing systems are defined as computing systems that can "*(1) learn models capturing knowledge about themselves and their environment on an ongoing basis and (2) reason using the models enabling them to act based on their knowledge and reasoning in accordance with higher-level goals, which may also be subject to change*".

Self-awareness mainly focuses on improving the system perception, which involves dynamic knowledge acquisition and utilisation to facilitate autonomous self-adaption. Specifically, in the context of self-awareness, how to use knowledge to adapt is usually termed as self-expression [49]. The self-expression process can deliberately plan and decide the next actions to allow the system to self-adapt its behaviour [143].

Effort in architecture-centric approaches has been made in engineering self-awareness in computing systems [49, 142]. In [49, 142], they provide a reference architecture for representing the system's design decisions and constraints. The architecture draws inspiration from human cognition and levels of awareness in the human brain to represent knowledge of each system component [142, 49]. As a result, the architectural framework extends the knowledge modelling component to a finer grain. From the perspective of knowledge, self-awareness can be categorised into five levels, namely stimulus-awareness, time-awareness,

goal-awareness, interaction-awareness and meta-self-awareness (see Figure 2.2) [49, 142]:

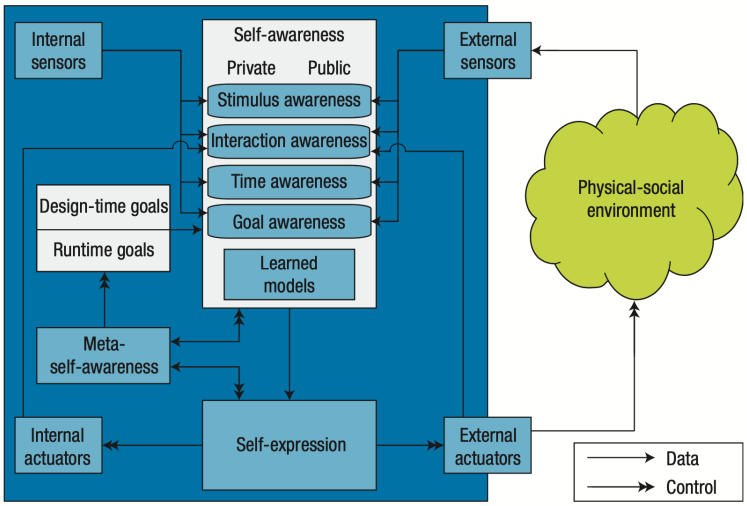


Figure 2.2: Reference architecture for self-aware and self-expressive computing systems (Figure from [142]).

- *Stimulus-awareness*: Stimulus-awareness is the basis for all levels of self-awareness and describes the simple reactive behaviour of the system. Stimulus-awareness is the ability to perceive the stimuli or events acting on the system and can consequently evoke the system to decide its next action.

- *Interaction-awareness*: An interaction-aware system can perceive historical knowledge of state, behaviour, and type of relationships and communication of interacting system components; it can consequently use this knowledge to decide its subsequent actions. Interaction-related knowledge may be represented as a graph or network among other representations.

- *Time-awareness*: Being aware of time means the system is able to use models that involve explicit memory, time-series modelling, or anticipation [142].

- *Goal-awareness*: Goal-awareness refers to the ability to know the system goals and reason about the goals. The awareness can be expressed as the utility or satisfaction of goals.

- *Meta-self-awareness*: The system can obtain knowledge of its own awareness levels and how they are exercised.

Different levels of self-awareness can exist within the system at the same time, but stimulus-awareness (i.e. the basic awareness level) is the prerequisite and enabler for other awareness levels.

The work in [73] further extends the self-awareness of individual systems to networked self-awareness. In the framework, the individual itself knows that other systems are also self-aware. The individual will be able to model and reason based on understanding the knowledge other systems will have, including their knowledge about stimuli, interaction, time, and goal.

2.2 Digital Twins

As an umbrella term, the specific definition of the "digital twin" varies in different research domains, but in general refers to the models that link with physical assets, objects, products or systems in real-time or throughout the full life-cycle of the physical entities. Although the Digital Twin is closely related to models and simulation, not all the literature stresses the role of simulation in a real-time data-driven manner [173]. For instance, some of the DT-related studies view the DT as a simulation environment that improves the design of a product in its full life cycle [230]. This perspective views the DT's role at a larger time span: data collection from the physical world and decisions made via a DT may only be ready after days or months. However, this thesis argues that DTs from this perspective are similar to offline simulation since the DT is not designed for real-time or near-real-time adaptation and decision support while the physical system is operating or rapidly evolving. This thesis only studies DTs for systems that operate in a highly dynamic physical environment and

that require "online" decision support. The "digital twin" should continuously receive data from its physical counterpart and the physical environment, and informs decision-making in a real-time manner [23, 149, 173].

2.2.1 A Brief History of Digital Twins

The initial idea of the Digital Twin emerged around 20 years ago, but only received wide research attention within the past 10 years. The DT concept stems from manufacturing and is now being applied to various application domains [231, 93, 161, 3]. The very first conceptual model of the DT was introduced by Michael Grieves in 2002 for product lifecycle management, but was not called a "digital twin" back then [94]. One of the earliest and most fundamental documented definitions of the DT was given by NASA in 2012 as an initiative for future generations of vehicles operating in extreme service conditions: "*A Digital Twin is an integrated multiphysics, multiscale, probabilistic simulation of an as-built vehicle or system that uses the best available physical models, sensor updates, fleet history, etc., to mirror the life of its corresponding flying twin* [93]." With the simulation predictive capability, the DT can forecast the system's health, remaining useful life, and system response in previously unknown conditions [93]. Tao et al. proposed a five-dimensional model of the DT in the manufacturing domain, which defines five components of the DT: physical part, virtual part, connection, data, and service [232, 231]. In [171], the authors further add to the definition of the DT with the capabilities of self-adapting, self-regulating, self-monitoring, and self-diagnosing. A taxonomy of DTs is also presented in [235]. Although there are different variations on the components of the DT, they all reflect the earliest conceptual idea by having the physical part, the virtual part, and the data connection between the two parts [1].

[1]Different existing literature may refer to the physical part as "physical object", "physical entity", etc. Also, the virtual part may be called "virtual twin", "digital part", etc. This thesis does not distinguish such differences.

The research emphasis of the DT has been recently put on the dynamic and bi-directional connection between the DT and the system being modelled, and how to link the physical object and digital object in an accurate and real-time manner [150]. In the Internet of Things (IoT) context, Minerva, Lee, and Crespi [173] proposes a set of essential properties that characterise the DT, among which "the constant entanglement between an artifact and its software representations" is of prominent importance to reflect changes of the physical artifact to the model in real-time and vice versa. The directions of data flow between the DT and the physical object/entity can be classified as virtual-to-physical (V2P) and physical-to-virtual (P2V) [233]. In P2V, the state of the model is updated with sensor data. In V2P, the up-to-date model is used to predict the future state of the physical system and actuate decisions back to the physical system.

Recent studies have further considered the composition of a DT-based system and the coordination of multiple digital twins. Digital Twin Networks (DTN) focus on the communication technology-enabled real-time information interchange among physical objects and virtual twins [260]. In [199], the notion of Web of Digital Twins (WoDT) is proposed to envision a distributed ecosystem of connected digital twins enabled by the widespread use of software in large-scale interconnected physical environments. This vision combines knowledge from the World Wide Web, the IoT, multi-agent systems, and distributed systems. It envisions a service-oriented software layer for possible intelligent applications to be designed. The concept of the Internet of Digital Twins (IoDT) stems from the IoT domain [251]. It focuses on a decentralised structure and the data and information exchange in *inter-twin* (between the physical entity and its digital twin) and *intra-twin* (between digital twins) modes.

2.3 Online and Lifelong/Continual Learning

Concepts similar to online knowledge enrichment involved in this thesis have been addressed by the research in online learning and lifelong/continual learning. This section introduces these concepts and their applications in self-adaptive systems. The differences from this thesis is also identified.

Machine learning research has actively explored *online learning*, a paradigm where the system learns from a sequence of data instances one by one. This is different from batch or offline machine learning methods, which usually aim to learn a model from the entire training dataset at once [35, 102]. Online learning can be applied to supervised learning, unsupervised learning and reinforcement learning. For reinforcement learning in particular, studies have also focused on combining offline static data into online reinforcement learning to minimise online interactions and learning efficiency [240, 20, 218].

The use of online learning has been explored in the research on self-adaptive systems. Since the self-adaptation logic needs to be defined at design time to handle uncertainties at runtime, the motivation for employing online learning is to rely on less the prior knowledge of the environment at runtime [114] and the difficulty for engineers to anticipate all possible runtime environment scenarios [189]. For instance, Jamshidi, Sharifloo, Pahl, Metzger, and Estrada [114] use fuzzy Q-Learning to achieve knowledge evolution of a self-learning cloud controller for the auto-scaling of compute resources at runtime. The adaptation rules of the controller are learned and dynamically modified during runtime. Palm, Metzger, and Pohl [189] use policy-based reinforcement learning and eliminates the need to manually quantise environment states and manually fine-tune the exploration rate. Metzger, Quinton, Mann, Baresi, and Pohl [167] propose novel system-evolution-aware strategies that utilise feature models to guide efficient exploration in online reinforcement learning for self-adaptation. In addition, the use of deep reinforcement learning, where knowledge is encoded into neural

networks, poses several challenges such as explainability [76], dynamic context shifts [8], that require further research.

Lifelong machine learning [50] refers to the ability of a system to learn continuously a sequence of different tasks by leveraging the past learned knowledge. It is coined to distinguish with classical machine learning that is only trained once with a static dataset for a single task without considering the past learned knowledge. Compared to lifelong machine learning, online learning only focuses on performing the same learning task over time [50]. *Continual Learning* is a concept often used interchangeably with *lifelong machine learning* [129, 50, 62]. As mentioned by [129], combining reinforcement learning with continual learning is a natural fit due to the similar concern for challenges in non-stationary environment dynamics, catastrophic forgetting [262], and prioritising recent and past experiences.

Preliminary works have utilised lifelong/continual learning in self-adaptation to tackle the challenge of continuously emerging tasks at runtime. One of the problems to be solved by lifelong self-adaptation is dealing with concept drift, a common challenge in applying machine learning in dynamically varying environments [253]. Chen [47] empirically examines two generally adopted methods for performance modelling to deal with concept drift in learning-based self-adaptation: retraining the model completely by combining the new data and all the historical data, and incremental learning using the newly arrived data sample. The author has provided evidence-based insights and statistically significant factors for the choice of modelling methods. Gheibi and Weyns [91, 90] study three different types of concept drift in lifelong self-adaptation: sudden covariance drift, incremental covariance drift, and the drift of adaptation spaces. The authors have proposed a general architecture and corresponding mechanisms to deal with new learning tasks at runtime. Masood, Jiangbin, Ahmad, Dongdong, Shabbir, and Irfan [164] propose a novel expert system based on continual reinforcement learning for self-optimisation, designed to evolve a knowledge base of adaptation rules that meet both user and quality objectives.

Despite the similarity between the idea of revising knowledge in online and lifelong learning and the idea of online knowledge enrichment in this thesis, online and lifelong learning assume that there is only one learning system, while this thesis considers the knowledge management in both the physical system and its DT. In lifelong self-adaptation, only the managing system evolves its knowledge without assuming the existence of dynamic knowledge management capabilities in the managed system. This thesis focuses on the novel problem scenario of having a knowledge-aware intelligent system and investigates the mutual knowledge enrichment relationship between the system and its DT. The DT is assumed to have more advanced intelligence than its physical twin system, which is supported by superior computational capabilities and simulation models. The use of a DT with online simulation capability for high-fidelity forecast and analysis is not commonly addressed in online and lifelong learning.

2.4 Related Concepts in Other Domains

There are concepts from other domains that resemble the idea of the DT. These concepts take different research perspectives. There are both similarities and differences with the idea of the DT. This section analyses the related research communities and compares the similarities and differences.

2.4.1 DDDAS/Info-Symbiotic Systems

A concept similar to the DT is Dynamic Data-Driven Applications Systems (DDDAS), which is first introduced by Frederica Darema in the early 2000s. DDDAS describes the ability to "incorporate additional data into an executing application (these data can be archival or collected online), and in reverse, the ability of applications to dynamically steer the

measurement process" [60]. The core of DDDAS is the data assimilation and the sensor reconfiguration computational feedback loops. Such feedback loops are computational rather than feedback control [34]. In feedback control, the input is the force or other physical properties. In contrast, computational feedback does not involve direct interaction with physical properties. The data assimilation loop dynamically incorporates extra data into the model, where errors between the simulation prediction and the sensor data from the physical system are used to refine the model, enabling the simulation to follow the trajectory of the physical system. In the sensor reconfiguration loop, the application simulation will, in return, adjust the sensing and measurement (e.g. sampling a subset of the entire measurement space) for better modelling and prediction purposes [34].

Although the Digital Twin and DDDAS share a similar idea of assimilating data continuously to the model for more informed adjustments, there are a few differences that distinguish the two concepts. The Digital Twin focuses on different engineering properties and poses new requirements. (1) The Digital Twin paradigm emphasises more on replication, and there must be a corresponding entity that exists in the physical world. The digital twin should maintain tight entanglement with the physical entity. The foundation of the Digital Twin is the real-time representation of the physical entity or process. Otherwise, it is not called a "twin". DDDAS focuses more on the feedback loop and how data is assimilated and used dynamically. It does not specifically require the explicit representation of the physical entity, although such a requirement may be implicitly involved in modelling. (2) As for the feedback loop that involves affecting the physical world, Digital Twins are not restricted to measurement reconfiguration as that in DDDAS. A DT can help adapt and optimise the behaviour of the physical system. A DT can act as a surrogate, as in a cyber-foraging system, to delegate computation for physical objects and provide additional knowledge to inform decision-making for the physical object.

Despite the differences, we regard a Digital Twin as essentially an enhanced DDDAS-

based system, as the Digital Twin paradigm also incorporates dynamic data into models in a feedback manner and influences the behaviour of the physical system. The DDDAS paradigm can be an enabling foundation for the DT in the design of feedback loops between the physical system and its models [66, 89, 120, 162].

2.4.2 Symbiotic Simulation

As described in [18], the term "symbiotic simulation systems" was coined by the parallel and distributed simulation working group at the Dagstuhl Seminar on Grand Challenges for Modeling and Simulation in 2002 [81]. Symbiotic simulation systems are regarded as a new paradigm for discrete event simulation, and the decision-making is assumed to be based on what-if analysis [81, 18]. It consists of the physical system and its simulation counterpart to form the symbiotic feedback relationship in between. Aydt, Turner, Cai, and Low [18] proposes five basic classes of symbiotic simulation systems. Onggo, Mustafee, Smart, Juan, and Molloy [187] extends the work and discusses research issues with hybrid modelling, big data analytics and machine learning.

While DDDAS and Symbiotic simulation are both data-driven, the symbiotic simulation system refers more specifically to data-driven simulation, while DDDAS refers to general applications [17]. Also, the symbiotic simulation system involves control feedback to the physical system. This feedback is meant to control the physical system, but not necessarily to only control the measurement process as in DDDAS [18]. But in general, DDDAS and symbiotic simulation share the same idea.

Similar to the discussion in the previous section regarding DDDAS and Digital Twins, the Digital Twin also overlaps with the symbiotic simulation, but provides more possibilities surrounding the replication of a physical *object* in finer grain and various added value for the

physical object such as augmentation, entanglement, composability, etc. as characterised by [173].

2.4.3 Parallel System

Another concept that is similar to the DT is Parallel System, which was coined by Fei-Yue Wang in 2004 [243]. The core of the Parallel System is the ACP approach, which informs parallel control and management of complex systems. The ACP approach is a conceptual framework that contains Artificial societies (A), Computational experiments (C), and Parallel execution (P) [242]. The approach assumes the complex system is inseparable – independent analyses of individual components cannot explain the behaviour of the entire system, and unpredictable. Artificial societies serve the need to represent the complex physical system in a holistic manner. Solutions for problems in the complex system are not fixed and should be adaptive with more experience that has been learned. Adaptive solutions are needed and can be solved through computational experiments. No optimal solution exist, therefore Parallel execution provides a way to evaluate the performance of various solutions [242].

The paradigm of parallel systems and the ACP approach have been applied to various domains, such as healthcare systems [247], robotics [269], smart traffic control [274], autonomous vehicles [146], etc. The idea of parallel systems has also been advanced towards ACP-based parallel intelligence [244, 248].

Both the DT and the parallel system method share the idea of representing a physical system and using what-if analysis to simulate the outcome or performance of different inputs to the physical system. Parallel System, in its original idea, focuses more on the social aspect. It focuses more on the human factor and the human's aim to guide the physical system. The

parallel system paradigm does not have the emphasis on the requirement for replicating a real-world system as that in the Digital Twin [263].

2.5 Research Perspective

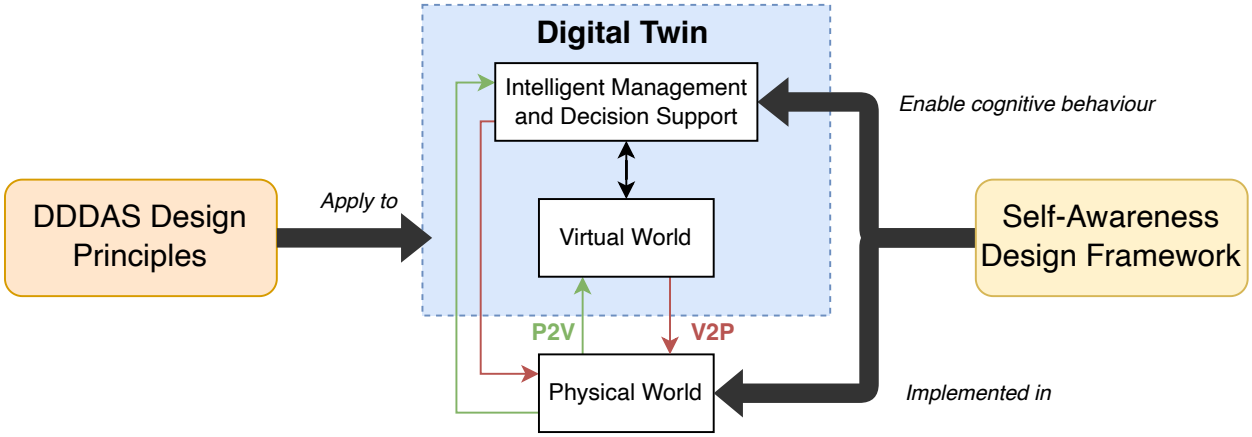


Figure 2.3: The conceptual model of the Digital Twin, Self-awareness, and DDDAS.

This thesis examines how existing work can be adapted to enable the cognitive behaviour of digital twins, and how to achieve mutual intelligence enrichment through P2V and V2P between the DT and its physical counterpart. Since self-awareness is closely related to cognition and knowledge, This thesis considers self-awareness as the fundamental enabler for the intelligent capabilities of a system. The intelligence of both the DT and the physical system can be architected according to self-awareness principles. In addition to intelligence, one essential dimension of the DT is the feedback loop between the physical world and the virtual world. This thesis applies the DDDAS principles as a solution to mutual intelligence enrichment, in order to address how knowledge is obtained, updated, and utilised in the feedback loop of the DT. Figure 2.3 shows the conceptual model adopted by this thesis, which describes the relationship between the DT, DDDAS, and self-awareness to be covered in the rest of the chapters.

Chapter Three

Digitally Twinning an Intelligent System

The basis of the DT paradigm is the replication of an existing system in the real world. It is important to define what is the "physical twin" discussed in this thesis. Specifically, it is imperative to define the components of the physical world and how these components are characterised. Also, since being self-aware is the main assumption of the thesis that the real-world system should be, it is important to clarify and define how self-awareness is manifested in the system. This chapter defines the real-world intelligent system considered in this thesis, and describes the digital twinning of such a system as well as the environment in which the system operates. Based on the twinning, a generic reference architecture is proposed for intelligent DTs of self-aware systems.

3.1 Related Work

Designing DTs for systems that are already embedded with intelligent self-* capabilities is still at a preliminary stage. Much of the work involving architecture/reference model design focuses on offloading the intelligence that can originally be run on the physical system to the DT. Efforts have been made in architecting DTs of autonomous robots with co-simulation

[156]. The work in [156] proposes an architecture to delineate the components specific to the control of robot applications. The decision-making capability of the robot is offloaded to the DT which supports movement prediction and emergency stops. There is also research work that studies collective digital twins for systems that are composed of collective nodes. The work in [205, 203] proposes frameworks for autonomic and cooperating digital twins with the integration of dynamic context management, runtime models, adaptive control, and feedback loops. The framework contains dynamic context management by the MAPE-K (Monitor-Analyze-Plan-Execute over a shared Knowledge) loop, which is widely adopted by self-adaptive systems. Each digital twin is able to run experiments to update its knowledge to achieve its goals, while all digital twins can self-organise for a cooperative reaction to a shared goal. Casadei et al. coined the notion of "augmented collective digital twin" for self-organising cyber-physical systems, which fuses the concept of the DT, virtual devices, and collective systems [44]. They have designed a meta-model that maps logical devices and their internal components to physical nodes, i.e. where the components of the logical device are deployed. The self-organising behaviour happens among the logical devices in the virtual space. In [175], the authors have proposed a framework for an intelligent DT to improve the situational awareness of mobile robots. They have designed metrics that enable assessing how well the robot is aware of its environment.

Apart from self-*, preliminary work has applied DTs to facilitate federated learning [225] and multi-agent learning [270], where the learning is conducted on IoT or edge devices. Although their DT can optimise the learning process with certain decision models, the DT does not involve the essential capability of simulation analysis. Simulation has been addressed as one of the most fundamental capabilities to support more sophisticated analysis and prediction [93, 214, 231]. The lack of simulation may impede the DT's ability to forecast and evaluate prescriptive what-if scenarios to make more informed decisions. The work in [249] allows reinforcement learning agents to learn in both the DT and the real world.

However, the DT is simply a simulation environment; it does not involve intelligence such as adaptive behaviours.

To summarise, most existing efforts in augmenting the self-* intelligent capabilities of the physical system have been put forth in offloading full intelligence to the DT. Table 3.1 compares this thesis with the most relevant papers on digitally twinning intelligent systems. An assumption not adequately explored is that a limited level (due to resource constraints) of intelligence and autonomy residing in the physical twin and the DT augments that intelligence with what-if simulation analysis. Existing literature also does not consider the situation where the real-world system explicitly has levelled knowledge management capabilities, which is a common design methodology for empowering full capabilities of self-awareness [142, 143, 26]. Further, the basis of accurate simulation is an equivalent model in the DT; when modelling physical systems that can self-learn, additional requirements should be raised to ensure the knowledge of the physical system is also equivalently maintained in the DT. Therefore, there is a need for a methodology for equivalently modelling (digitally twinning) an intelligence system. One of the contributions of this thesis is a reference architecture that addresses the above-mentioned problem of knowledge equivalence, which is not adequately discussed in the existing literature.

Table 3.1: Related work for digitally twinning an intelligent system.

Ref.	**Objective**	**Real system**				**Digital Twin**	
		Type of intelligence	Distributed intelligence	Explicit knowledge management	Fine-grained runtime knowledge	Simulation analysis	Replicating intelligence of the real system
[156]	Monitoring and prediction	Behaviour controlled through teleoperation by human users.				✓	
[175]	Improving situational awareness	Situational awareness		✓	✓		
[225]	Efficient training	Federated learning	✓	✓			
[270]	Efficient training	Multi-agent learning	✓	✓			
[249]	Improve learning efficiency	Reinforcement learning		✓		✓	✓
This thesis	Mutual knowledge enrichment	Fine-grained online knowledge management	✓	✓	✓	✓	✓

3.2 Digital Twinning

3.2.1 Physical World Model

This thesis separates the physical world into two parts: the *physical environment* and the *physical system*. The behaviour of the physical system is assumed to be regulated by software programs that have self-* capabilities, thus being an intelligent system. The physical system contains both the self-* software programs and the managed software/hardware that is directly controlled by the self-* part. Here the *managed* part that is controlled by the self-* software program is similar to the idea of the *managed system* as in [256]: the managed system can be either hardware or software. The physical environment is the part that cannot be controlled by the software but is only affected by the behaviour of the physical system.

This thesis considers the physical system to be composed of computing nodes with embedded intelligence. Each node is controlled by an onboard intelligent self-aware agent (referred to as a *physical agent*) with its own local knowledge about itself and the environment. Agents collaborate and accumulate knowledge as shown in Fig. 3.1. Such a configuration is common in the studies of decentralised self-adaptive/self-aware systems [257, 194]. For instance, in surveillance applications, geologically distributed smart cameras can be deployed to track objects of interest as they move through a given space, such as a city street. The cameras self-organise the tracking handover without central control or prior knowledge of the camera network topology. Each camera, as a computing node, is embedded with an agent that communicates with other cameras to negotiate the tracking responsibility. Each agent can be designed with time-awareness and interaction-awareness: adaptively learning knowledge of interaction from successful handovers with other cameras in the past to infer the camera network topology, which can be used for handover negotiations [75]. In this example, the agents and the handover behaviour of the cameras constitute the physical system. The

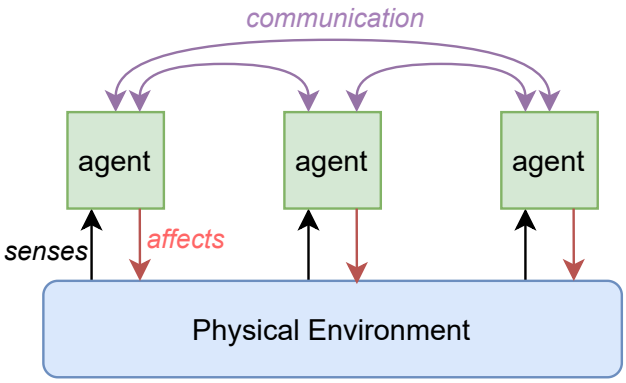


Figure 3.1: Physical world.

moving objects, the city space, and other parts of the camera not controlled by the agents (e.g. battery) are regarded as the physical environment. Another example is decentralised elastic data stream processing in the cloud [42], where operator programs running on server nodes need to process incoming data streams in multiple steps to get the output. To speed up the processing while minimising reconfiguration costs, each operator instance can be scaled in/out and migrated to other nodes. Each operator can be managed by a self-aware agent individually. Each agent adaptively learns the resource utilisation pattern and decides the scaling and migration of each operator autonomously, which is more scalable compared to a centralised control scheme. In this example, since agents directly control the replication behaviour of the operators, the agents and the scaling behaviour of the operator programs are regarded as the "physical" system. The actual computation of the operator is beyond the control of the agents and thus is regarded as part of the physical environment. The server nodes and incoming data flow are also parts of the physical environment.

3.2.1.1 Intelligent Self-Aware Agents

The physical agents are assumed to possess self-awareness capability. An agent function is a mapping from perception to action [209, p. 35]. Inspired by the concept of *learning agents* [209, pp. 54–57] and *self-awareness* [142], this thesis further formally specifies the

intermediate process between perception and action of an agent by explicitly incorporating the awareness and utilisation of knowledge.

This thesis regards an intelligent physical agent α as a tuple $\{A, P, K, \phi, \pi\}$, where:

- A is a set of all allowed actions of the agent. The agent affects the environment via actions. Communication between agents is also viewed as actions [259].
- P is the agent's perception space of its environment and itself. The perception at time t is denoted as $p_t \in P$, which consists of processed data from the sensors. Example content of the perception data may include the received messages from other agents or the positions of the agent's surrounding objects in the environment.
- K is the space of the agent's runtime knowledge, with $k_t \in K$ signifying the agent's knowledge base at time t; Knowledge is represented according to the *computational self-awareness* architecture, explained in section 2.1.2.
- ϕ is the agent's awareness function, which constructs the agent's runtime knowledge k_t based on the perception p_t and its previous knowledge k_{t-1} (supposing time is discrete). ϕ can be used to instantiate the learning process in which p_t is the feedback after utilising k_{t-1}.

  $$k_t = \phi(p_t, k_{t-1}) \tag{3.1}$$

  It is worth noting that if the agent is only stimulus-aware, its awareness function ϕ will degenerate from $\phi : P \times K \to K$ as specified by equation (3.1) to $\phi : P \to K$, because stimulus-awareness alone is purely reactive and cannot preserve history.
- π is the agent's decision function. An agent utilises π to decide the next action a_t based on its perception p_t and current knowledge k_t.

  $$\pi : P \times K \to A,\ a_t = \pi(p_t, k_t) \tag{3.2}$$

Notice that the output a_t can be a single action or a probability distribution of different actions, which corresponds to deterministic and probabilistic decision-making, respectively.

The agent follows a sense-think-act cycle. During *sensing*, the agent first obtains its perception p_t by sensing its environment and itself, including checking the messages it receives from other agents (e.g. through a mailbox [46]). In the *think* process, the agent uses the current p_t as feedback to update its knowledge by ϕ. Then π will derive an action or a probability distribution of actions. Finally, at the *act* stage, the action a_t is executed through actuators to affect the environment or/and other agents.

Knowledge is different from an internal "state". A state within an agent is composed of a set of attributes that represents the current status of the agent or the current representation of the world. Examples of the internal state of an agent can be the energy consumption level, the current action, the agent's location in the environment, etc. However, knowledge refers to the set of beliefs and information that an agent has accumulated over time and that the agent uses to make decisions or take actions. It is often represented by a set of rules, models, or beliefs that describe relationships between variables and that are used by the agent to reason about its environment. These rules, models, or beliefs can be revised as the agent continuously interacts with the environment, which refers to the notion of online learning. Examples of knowledge may include the model of the environment dynamics, goals and how to achieve goals, perceived interaction patterns with other agents, etc. Knowledge supports the mapping from state to action.

Fig. 3.2 illustrates a self-aware agent based on the formal definition above and the self-awareness paradigm outlined in section 2.1.2. The self-awareness component constructs and modifies the knowledge base by its capabilities at one or multiple level(s) with sensor perceptions. The *self-expression* component expresses decisions and yields consequent

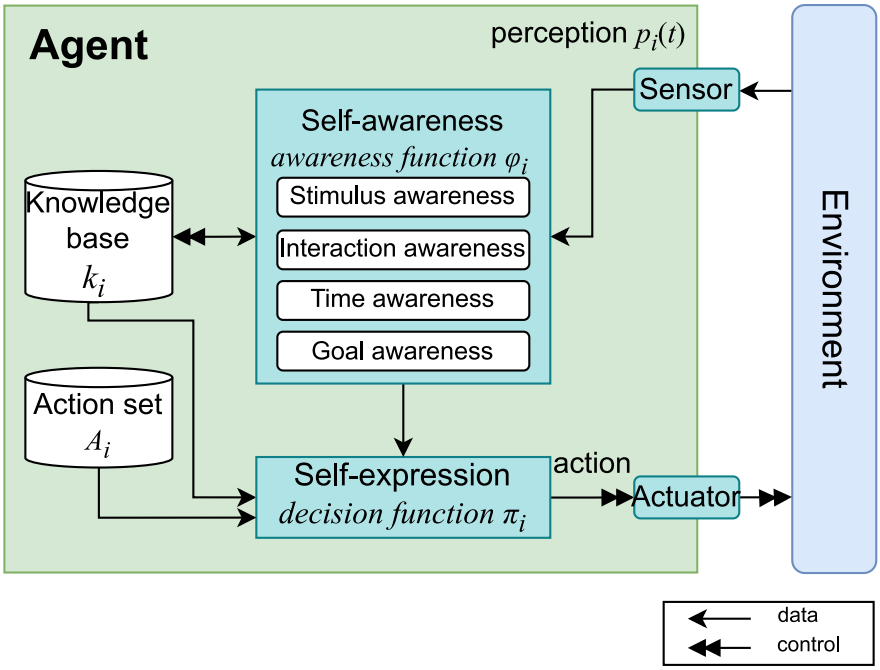


Figure 3.2: A self-aware agent in the physical world.

actions entailed by the awareness level(s).

3.2.2 Virtual World

The Digital Twinning of the physical world involves replicating agents and the physical environment as *virtual agents* and *virtual environment*, as shown in Fig. 3.3. For the clarity of terminology, throughout the rest of the thesis, the system that operates in the physical environment is called the *physical system*[1]. A physical system is composed of agents, which are referred to as *physical agents*. Correspondingly, the DT model of a physical system is called a *virtual system*. The model of a physical agent is a *virtual agent*. The model of a physical environment is called a *virtual environment*. The modelling scheme for the behaviour of agents and the dynamics of the environment are as follows.

[1]This thesis will use the terms "physical system" and "real-world system" interchangeably.

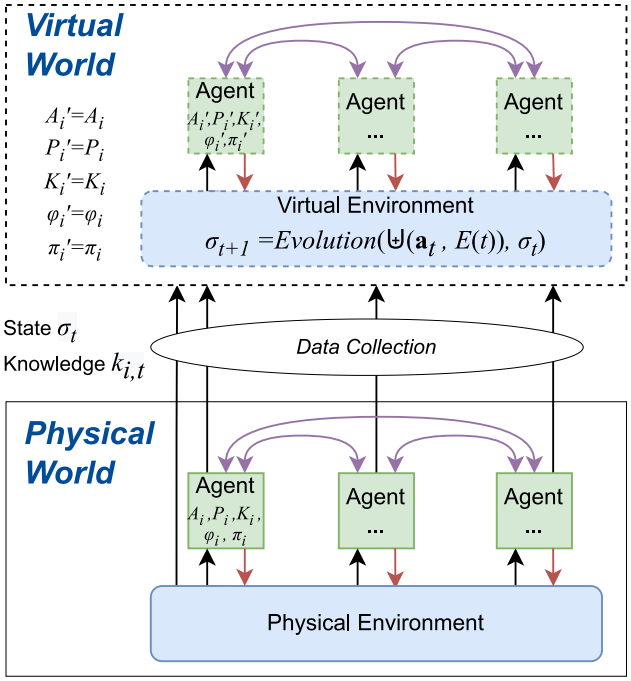


Figure 3.3: Digital Twin modelling of the physical world.

3.2.2.1 The Agent Model

Since an agent is essentially a software system, the modelling towards a virtual agent can be realised by replicating the same software program of the physical agent. Specifically, a virtual agent possesses the same $\{A, P, K, \phi, \pi\}$ as its physical agent counterpart, which is also shown in Figure 3.3.

3.2.2.2 The Environment Model

To model the environment, this thesis takes a state-based perspective [138, 227]. The environment is considered as a two-tuple: $Environment =< State, Process >$. The environment has its own endogenous dynamics denoted by $Process$ that can change its $State$, independent of the actions of agents. Let $\sigma \in \Sigma$ be the state of the environment. At any time t, the

state of the environment σ_t can be characterised by a vector that contains multiple variables:

$$\sigma_t := (\sigma_{1,t}, \sigma_{2,t}, ...\sigma_{k,t}) \tag{3.3}$$

For instance, in a spatial context, σ_t can be composed of the positions of all physical entities within the space, such as physical agents, obstacles, etc. Variable $\sigma_{i,t}$ represents the position of the i-th entity.

The environment is affected by the agents and also evolves according to its own endogenous dynamics. The coupling of the environment and agents results in the overall world dynamics, which is defined by a function *Evolution* such that the transition from current moment t to the next time step $t+1$ is the composite result of all the agent actions and the endogenous dynamics of the environment [170]:

$$\sigma_{t+1} = Evolution(\uplus(\mathbf{a}_t, E_n(t)), \sigma_t) \tag{3.4}$$

$\mathbf{a}_t = (a_{1,t}, a_{2,t}, ..., a_{m,t})$ is all the action made by each agent onto the environment at time t, and $a_{i,t}$ is the action made by agent i at time t. $E_n(t)$ is the dynamics produced by the natural evolution of the environment (the *Process*). The symbol $\uplus$ denotes the action composition operator, which defines how the actions and endogenous dynamics of the environment at t are composed to calculate the consequences on the previous environment state σ_t [170].

3.3 A Generic Reference Architecture

Based on the model abstractions in the previous section, where the physical system is an intelligent multi-agent system, a reference model of the DT is proposed in Fig. 3.4. The DT and physical world interact through a communication layer. The data (perception and

knowledge) sent from the agents in the physical world are first used to create the real-time virtual replica. Then, these data are used to drive the model equivalence and decision support.

3.3.1 Cognitive Decision Support

In the reference model, a Cognitive Decision Maker is introduced, whose design can be enriched by human cognitive behaviour of self-awareness to support predictive and prescriptive analysis. The detailed design of a self-aware cognitive decision-maker will be discussed later in Chapter 4. The cognitive decision maker monitors any event that is causing or may lead to sub-optimality of the physical world and then provides more informed decisions to optimise the behaviour of the physical system. The analysis towards decisions is enabled by a library of analysis tools, among which the what-if analysis is of paramount importance. With what-if analysis, the cognitive decision maker can obtain an in-depth understanding and explanation of the mechanisms governing the system's behaviour through simulation. By exploring different scenarios or experimenting with alternative decisions, the cognitive decision maker can deliver actionable insights suggesting the best solution given a variety of choices.

The decision-making for the intelligent multi-agent system is performed at a meta-level. Since agents already have a certain level of intelligence and autonomy as described in Section 3.2.1.1, the DT mainly *assists* the intelligence of the agents, namely, on adapting the agent's decision function π, revising and enriching the knowledge base k_t, or altering the awareness process ϕ.

For the DT-assisted adaptation of the decision function, an agent relies on its $\pi \in \Pi$ to derive an action, where Π contains all possible decision functions that this agent can

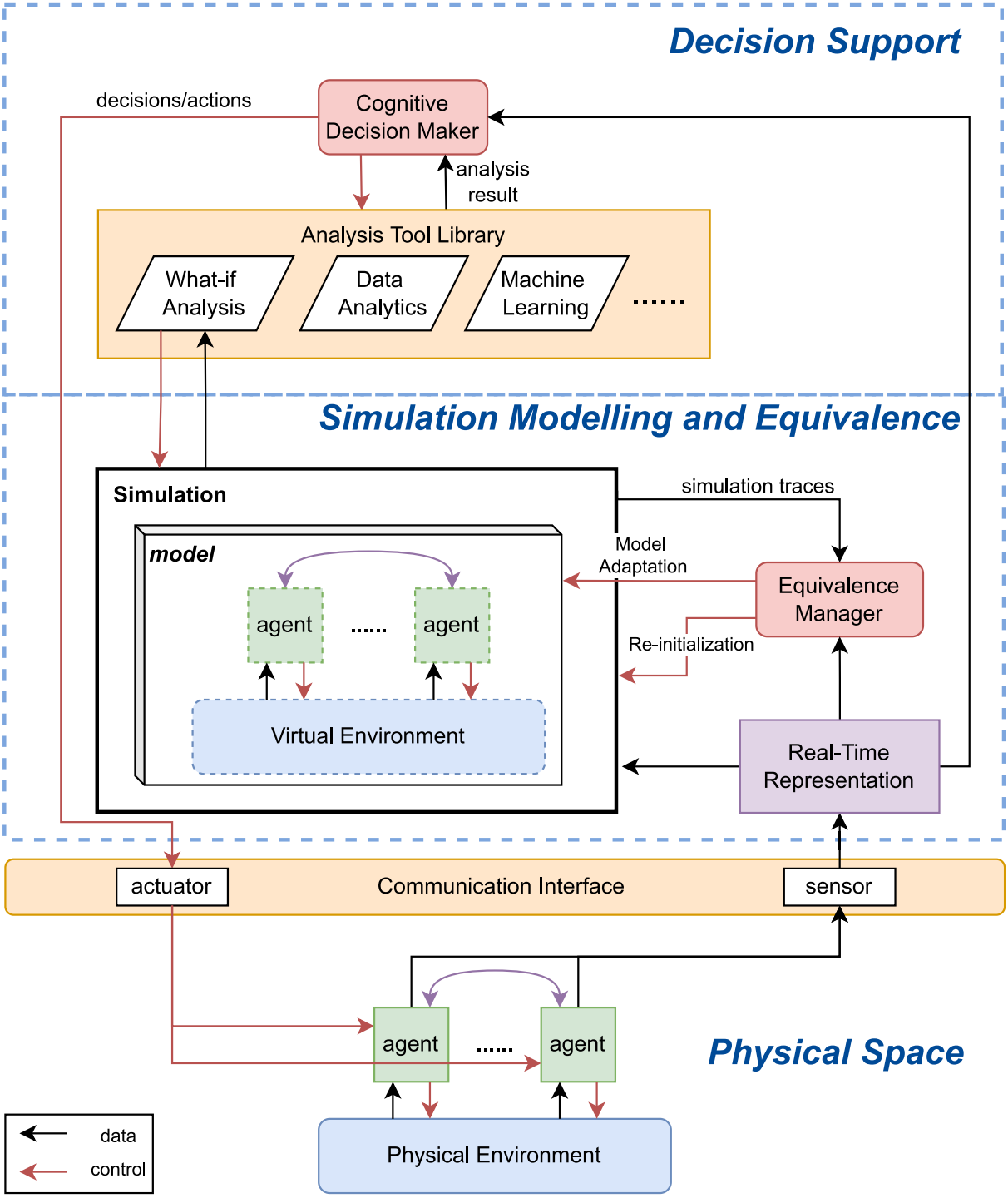


Figure 3.4: A reference model of the Digital Twin for intelligent systems (agents are self-aware).

be equipped with. The digital twin can dynamically select one $\pi_i \in \Pi$ that is most suitable for different scenarios. For instance, consider two choices of the decision function: a condition-action rules set and a deep neural network. Condition-action rules are reliable but less adaptive to new specialised conditions not listed in the rules. While a deep neural network is robust for handling scenarios that are not encountered during offline training, the online computational cost may be higher than matching the rules, thus increasing energy consumption. The neural network, as a black box, may also lack explainability and trustworthiness. Dynamic switching between the above two choices can enable more flexible reactive abilities for the agents to operate in a complex physical world, but directly applying such a switch may risk degrading the physical environment. With a DT, the outcome of different candidates of π can be evaluated in the simulation before enacting any changes in the physical world. Another example of switching π can refer to deciding what levels or dimensions of knowledge to utilise. The knowledge base of the agent may be composed of different levels of knowledge about its stimulus, interaction, goals, etc. Different $\pi \in \Pi$ may utilise different levels of knowledge in order to derive an action. By switching the level of knowledge, the DT can serve as an instantiation for the notion of meta-self-awareness [69].

At a meta-level, the DT can also assist in the adaptation of the knowledge base k_t and the awareness process ϕ of the physical agents. The DT can provide more knowledge that cannot be obtained by a single agent. Agents may only have a limited perception of the entire world, due to constraints on the sensor, processing capabilities, etc. The DT can send knowledge of the entire world to each agent by gathering and analysing data and information perceived by all the agents in their own history. An example of such knowledge can be the people density distribution and moving patterns of the entire city street in city surveillance applications. The DT can also use simulation to allow agents to experience different scenarios that might happen in the future, thus facilitating agents to speed up the learning cycle enabled by the awareness process ϕ and accumulate knowledge faster.

3.3.2 Model Equivalence and Adaptation

The fundamental requirement for the DT is to ensure that the model (virtual world) is equivalent to the physical world. To achieve equivalence, the DT needs to collect data continuously from the physical world and check whether the model is a valid representation of the physical world. Data describing the agents and the environment are both needed. Each agent α_i should push its perception $p_{i,t}$ (or part thereof) and current knowledge $k_{i,t}$ to the virtual world, such that each virtual agent is updated with the most recent perception and knowledge. To obtain data describing the physical environment, dedicated IoT sensors are required to capture the state of the environment. However, if no such IoT sensors are deployed and agents are the only entities in the physical world that are able to transmit data to the virtual world, the current environment state σ_t can be interpreted by combining the perception from each agent. An Equivalence Manager component is introduced to maintain an equivalent model in a cost-efficient manner. The equivalence manager continuously compares the model against the physical world and, in case of discrepancies, initiates additional data assimilation or adaptation of the model [127]. The cost-efficient equivalence maintenance is discussed later in Chapter 5.

Chapter Four

Architecting Intelligent Digital Twins with Self-Awareness

Based on the generic reference architecture proposed in the previous chapter, this chapter designs an intelligent Digital Twin by refining the architecture. Existing related efforts in engineering intelligent digital twins are reviewed and classified based on their intelligence capabilities in different dimensions. This chapter specifically peeks into the detailed design of the Cognitive Decision Maker in the generic architecture (see Figure 3.4) from a knowledge perspective. This chapter enriches the design of intelligence by mapping self-awareness to the Digital Twin. The Cognitive Decision Maker is refined with the self-awareness framework and simulation capabilities.

4.1 Overview

This chapter takes the position that incorporating intelligence and cognition in a digital twin is a prerequisite to unlock the full potential of this disruptive technology, and to provide seamless integration, calibration and info-symbiotic collaboration between the two worlds

– the physical and its virtual replica. By intelligence and cognition, it can be argued that the "twining" property should be harnessed and enriched by mental and self-awareness-like capabilities and processes that leverage dynamic knowledge acquisition, management and its synergy across the digital and physical worlds, covering dimensions that relate to stimuli, time, interaction, goals, and context/situation-switching. Part of the objective is to promote self-adaptive analysis and proactive/intelligent decision-making/updates in an info-symbiotic manner between the two worlds and to operate on a richer and finer-grained knowledge base. Systems characterised by openness, dynamism and uncertainty in their operations, such as Cyber-Physical Systems and socio-technical environments, can particularly benefit from this vision. Additionally, cognitive digital twins (CDTs) can act as a Cyber Foraging environment, serving as "surrogates" for intelligent analysis, planning and simulation that would be computationally expensive and extremely difficult to carry out in real-time in environments that are characterised by limited computational power and capacity. This chapter endeavours to articulate the vision of the Self-Aware Digital Twin. This chapter introduces a reference model that builds on theories in human self-awareness in psychology and cognitive science to provide a reference architecture model for the Self-Aware Digital Twin. This chapter discusses various levels of self-awareness in cognitive digital twins.

The novelty of the proposed reference model is to harness the Digital Twin with knowledge to enable new modalities for intelligence. The aim is to implement self-awareness knowledge into fine grains and map five levels of knowledge to envision the Self-Aware Digital Twin. Digital twins should be able to leverage continuous interaction between the two worlds to perfect the knowledge in both worlds.

4.2 Intelligent Digital Twins

To identify the essential characteristics of an intelligent DT, this section reviews seminal research works in designing intelligent DTs and classifies them by the level of intelligence from different dimensions. Such a classification can identify the gap towards a more enriched design for the intelligent capabilities of a DT.

The intelligence of DTs has been discussed as a vision in the literature. The work in [23] identifies the main characteristics of the DT; these include the use of ontologies, high-dimensional data-(de)coding and analysis techniques, closed-loop optimisation self-adaptation for resembling the physical twin during its whole lifecycle, predictive analytics, prescriptive analytics, and simulation "what-if". In [103], the authors take a higher-level view and propose three properties of a future DT: context-awareness, autonomy, and adaptivity. Further, the authors in [185] delineate the key capabilities of *adaptive digital twins* to be real-time simulation, online optimisation, and adaptivity. In DTs for the Industrial Internet of Things (IIoT), according to [261], the intelligence of the DT consists of three key features: adaptive resource allocation, self-awareness and evolution, and predictive maintenance.

The enabling of intelligence may also involve artificial intelligence (AI) techniques for the optimisation and planning purposes towards an autonomous system [15]. There are existing literature surveys that address the integration between AI and DT. In [219], the authors suggest that machine learning (ML) can be an essential approach to enable the "self-evolving" of the DT to improve itself. However, according to [219], using real-time ML as part of the DT feedback loop is still in the preliminary stages. A review by reference [197] explores how big data, AI, ML, and IoT technologies contribute to the development of digital twins. However, the literature reviewed is only categorised by industrial domains, and the discussion regarding the classification of intelligent capabilities is insufficient.

One of the most relevant survey papers is [261], which classifies the existing AI solutions for DTs in the context of the IIoT. A three-dimensional classification model is proposed, and the existing work is classified and mapped according to the following three dimensions (categories in brackets): AI Tools (*Machine Learning* or *Deep Learning*), AI Tasks (*Automation*, *Classification*, or *Prediction*), Application (*Industry*, *Transportation*, *Energy*, or *Others*). This thesis regards AI Tasks as a dimension depicting the intelligence of the DT. However, their classification is specific to the application of AI techniques, not intelligence in general. The three categories of AI Tasks may not be mutually exclusive, e.g. a DT may have both the capability of classifying anomalies and predicting the anomalies in the future, which may not be adequately classified according to this scheme.

The current literature is inadequate in holistically analysing the different levels and aspects of intelligence that have been addressed in existing work. To address this, this thesis classifies the intelligence of the DT into different dimensions and discusses the levels along each dimension.

4.2.1 Levels of Intelligence

Some existing work has made initial attempts to classify the levels of intelligence of a digital twin. Gabor, Belzner, Kiermeier, Beck, and Neitz [83] classify the control of the DT into four levels: physical necessity, machine-environment interface, immediate reaction, and planned reaction. Bauer, Oliveira Antonino, and Kuhn [24] discuss four evolution stages towards a sophisticated digital twin: offline monitoring and reactive control, online control, predictive maintenance, and proactive control.

To provide an overview of the levels of intelligence for the existing DT-related research, this thesis first categorises the intelligence capabilities into three specific dimensions:

autonomy, learning and knowledge management, and analysis capabilities. Then, the levels of intelligence are discussed along each dimension. The three-dimensional classification is shown in Figure 4.1. The level of intelligence increases along each of the three axes. A digital twin possessing high levels of intelligence in all three dimensions is denoted as a Cognitive Digital Twin. The rest of the section will discuss in detail each of the three dimensions and the related work.

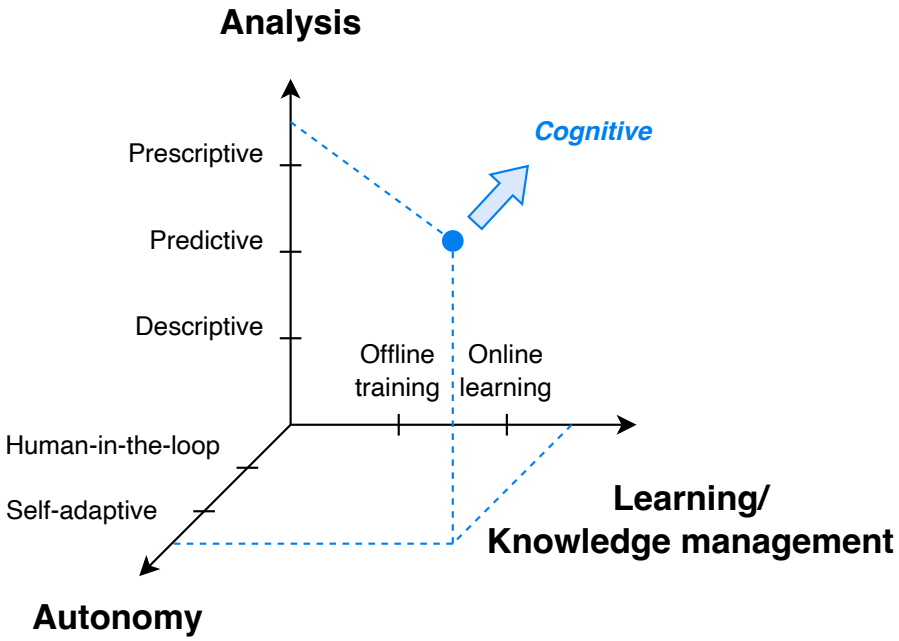


Figure 4.1: A classification of Digital Twins by intelligence.

4.2.2 Autonomy

The dimension of autonomy refers to the ability of a DT to form a real-time closed loop with the real world without human intervention. Digital twins that require a much longer time span for data to be fed back to the real system, such as in product design [230], are out of the scope of this thesis.

4.2.2.1 Human-In-The-Loop Control

For this class of intelligent digital twins, the purpose is to optimise the real-world system, but the analysing, planning and decision-making require human operators. Humans are the enabling factor for the intelligence of the digital twin.

Humans can be involved in simulating what-if scenarios and forecasting future possible incidents and events. Human-in-the-loop with simulation analysis has been applied to various domains such as management of logistics systems [134], security operation centre [67], hospital management [121, 122], aquaponics [3], etc. For security, human operators can simulate possible security attacks such as man-in-the-middle attacks [67]. For hospital management, patients' pathways can be monitored and predicted to inform decision-making, such as selecting the right health care [121, 122].

Further, humans can incorporate preferences or goals into the DT system and allow it to reach that goal by itself. In [265], humans interact with a Virtual Reality (VR) interface to specify the goal state of a robot; the robot will then use AI planners to reach the goal state.

4.2.2.2 Self-Adaptive Digital Twins

A self-adaptive DT does not need humans to make decisions. One of the common ways of designing self-adaptive DTs is to apply MAPE-K [77, 192]. Rivera, Jimenez, Villegas, Tamura, and Muller [204] proposed a reference model for software-intensive Digital Twins. They focus on applying MAPE-K feedback loops to evolve the runtime models and allow simulation and optimisation for configuration alternatives. DTs can be integrated with MAPE-K to provide various functionalities, such as self-healing and resilience [78, 25], explanations [265, 223], anomaly detection [28]. However, the research in applying MAPE-K to enable the

self-adaptivity of the DT is still in its infancy, with most of the research work focusing on design and proof-of-concept prototyping.

A self-adaptive DT can not only modify the behaviour of the real-world system but also change its own planning and decision-making. In [64], the authors enable the DT to dynamically switch between different decision-making models for the self-optimisation of mobile networks. They allow the DT to formulate decisions based on both expert knowledge and reinforcement learning, resulting in two choices. Then, the choice that leads to a higher reward in real-time simulation evaluation is applied to the mobile network. The DT then evaluates the two choices in real-time simulation and applies the one that leads to higher rewards to the mobile network. In [41], the authors use multi-fidelity models to support near-real-time decision-making. A low-fidelity ML-based model is used first to obtain results quickly and then guides more detailed evaluation using high-fidelity simulation models.

4.2.3 Analysis

According to the analysis that a DT can support, three levels of DTs can be distinguished: descriptive, predictive, and prescriptive.

- *Descriptive Digital Twin*: This level of analysis provides information about the world as it is, and was, but cannot tell much about the world as it might be.
- *Predictive Digital Twin*: Digital twins at this level typically utilise predictive data analytics such as data mining, statistical techniques and machine learning to extrapolate what might happen in the future. Such a digital twin only allows reliable extrapolation to scenarios similar to those that produced the historical or training data.
- *Prescriptive Digital Twin.* A prescriptive digital twin is able to utilise simulation and support what-if analysis, thus enabling explanations and a deep understanding of

mechanisms governing system behaviour. The exploration of multiple what-if scenarios delivers actionable insights suggesting the best solution given a variety of choices and optimising the result of future events or risks.

4.2.3.1 Descriptive Digital Twin

In some of the literature, the DT is viewed as an approach that enables real-time collecting of information about the physical world. These DTs are essentially *descriptive*. They can only provide information about the status of the system now or in the past, but cannot make any forecast about its future. The descriptive DT lacks models that can forecast or simulate the behaviour of the real world. For instance, in [270], the DT is used to facilitate multi-agent learning in vehicular edge computing. The DT tracks the vehicle states, including communication topology and computing resource demands. It dynamically aggregates agents into groups to improve their learning efficiency. However, the DT does not involve any predictive analysis that forecasts the future of the agents or vehicles. The aggregation decision is solely based on a reactive decision-making model and current data. Therefore, it is essentially a descriptive DT. In [225], the authors studied DT-assisted Federated Learning. The DT in this work is also descriptive, since the DT only provides the current status of the training node: current training state (loss function), CPU frequency, and energy consumption. No prediction is involved in the DT.

The essence of a descriptive DT is the monitoring capability. Using the DT to monitor the real-world entity, system, or environment has been addressed in various domains [202, 52, 58, 148]. The information to be presented by the DT can be *observable*, such as locations [10], or *unobservable*, such as structural health [58, 120]. For unobservable properties, one typical reason for the unobservability is the lack or incapability of sensors. For instance, it is sometimes preferable to monitor a production system with non-invasive instrumentation

and telemetry techniques [52]. A DT, in this case, can use physics or ML-based models to estimate the internal state of the system based on available sensor data. Another reason for unobservability is that the properties of interests are characterised by parameters that cannot be sensed by nature. One typical example is structural health, which is defined by parameters or coefficients in physics-based models [58, 120]. In addition, some of the work also uses ML models as a descriptive DT to estimate the state of the system (e.g. classify the current damage type [201]). One of the main reasons is that traditional physics-based simulation is computation-intensive and requires a large amount of time to obtain the results. Therefore, ML is used as a surrogate to substitute physics-based models for timely estimation.

4.2.3.2 Predictive Digital Twin

A predictive Digital Twin can forecast the future of the real-world system or environment. Predictive maintenance is one of the common application domains in which a predictive digital twin is involved. The digital twin can predict the remaining useful life to decide when to schedule maintenance in the future [236].

4.2.3.3 Prescriptive Digital Twin

A prescriptive Digital Twin takes a step further than a predictive one by supporting the evaluation of different "what-if" scenarios and decision choices by simulation. As early as 2014, Roßmann, Guiffo Kaigom, Atorf, Rast, Grinshpun, and Schlette [207] used a 3D simulation model as the mental model for the path planning of a mobile robot. The robot simulates all the possible future paths based on different initial parameters. In [63], the authors use discrete event simulation and multi-objective optimisation to optimise a production plant. Although conventional discrete-event simulation can be relatively time-consuming, its com-

putation speed can be significantly increased by parallel and distributed execution techniques to support real-time or near-real-time decision-making [82, 182].

There have also been many investigations into using ML models to support real-time what-if analysis. Naing, Cai, Hu, Wu, and Yu [179] uses Long Short-Term Memory (LSTM) to model the car following behaviours in city traffic. They use online learning to update the LSTM for more accurate prediction of the car behaviours. It supports just-in-time what-if simulation for short-time forecasts. In [245], the authors use an offline-trained Graph Neural Network (GNN) model to support network management. Various what-if scenarios can be evaluated by the GNN model to evaluate network performance with quality of service constraints. However, ML models can have various shortcomings compared to conventional simulation models (e.g. discrete event simulation models). ML models highly rely on large quantities of high-quality training data. However, data may not always be available since this requires costly telemetry systems to be installed in the real system [9]. ML models may also have the challenge of generalisation to accurately predict scenarios that are not seen in the training data [9]. Also, compared to conventional simulation models, ML models are black-box and not explainable.

4.2.4 Learning and Knowledge Management

This dimension refers to the DT's ability to acquire, update, and utilise knowledge for more informed decision-making. This dimension specifically focuses on how DTs can learn knowledge from *data* to support *decision-making*.

4.2.4.1 Offline Training

One of the common schemes of obtaining knowledge from large amounts of data is to train ML models offline and use these trained models for real-time decision support. This thesis regards this as the basic level of learning ability in this dimension of intelligence. For instance, reference [119] uses classification trees to infer the damage condition of UAV wings. The goal is to instruct the UAV to take different routes, considering the structural health. The work in [221] uses DTs to train deep reinforcement learning for the flocking motion control of distributed unmanned aerial vehicles (UAVs). These trained models can be used to estimate the current state of the system (descriptive digital twins), make predictions or support what-if analysis [245], or to directly control the real-world system [133, 139].

4.2.4.2 Online Learning

Online learning is a higher level of intelligence capability compared to offline training. It refers to the DT's ability to dynamically acquire, update, and utilise knowledge with data being collected in real time. Research at this level is still in its infancy. In [179], the authors use online supervised learning to update the parameters of an LSTM model that is used for traffic simulation. [222] uses online reinforcement learning for the control of wind turbines. [56] uses a default decision-making policy as a safeguard and a teacher, to continuously improve the policy of a reinforcement learning planner.

Some of the work focuses on using knowledge graphs to represent the knowledge possessed by the DT [210]. These knowledge graphs can be dynamically updated, and changes in the relationships of physical entities can be reflected in the knowledge graph [252, 6, 271]. For instance, [152] focus on using knowledge graphs to model the manufacturing resources and processes. They use a memorising-forgetting model to dynamically update the

knowledge graph.

To summarise, most of the existing work studies the dynamic update of knowledge graphs or black-box ML models. The knowledge graph mainly describes the relationship between entities. Knowledge in ML models is usually implicit and lacks interpretability. Research is still inadequate in addressing dynamic knowledge management explicitly at finer grains for intelligent DTs.

4.2.5 Cognitive Digital Twins

It is intuitive that intelligence can involve human cognitive behaviour. There is also a trend to raise the intelligence of the DT to the cognitive level by incorporating human cognitive behaviour into digital twins. Back in 2016, Gabor, Belzner, Kiermeier, Beck, and Neitz [83] viewed the digital twin as a virtual environment in which an external cognitive system can simulate faster than real-time to obtain insight for actions before applying it to the physical system. In the same year, Adl [2] also presented a vision for cognitive digital twins (CDTs): "*The Cognitive Digital Twin is a digital representation, augmentation and intelligent companion of its physical Twin as a whole including its subsystems and across all of its life cycles and evolution phases.*" In [7], the authors follow cognitive science and propose the cognitive capabilities of a DT: Perception, Attention, Memory, Reasoning, Problem-solving, Learning. The related work in Cognitive Digital Twins has been reviewed in [272], where the authors have identified common characteristics of cognitive DTs among past studies: DT-based, cognition capability, full lifecycle management, autonomy capability, and continuous evolving. They define a Cognitive Digital Twin as "*a digital representation of a physical system that is augmented with certain cognitive capabilities and support to execute autonomous activities; comprises a set of semantically interlinked digital models related to different lifecycle phases of the physical system including its subsystems and components; and evolves continuously*

with the physical system across the entire lifecycle".

One aspect of cognition in the DT is knowledge. Many of the studies capture the knowledge in the semantic-related aspects with ontology [57], where the real-world entities and their relationships are modelled. In [208], the authors use a unified ontology to describe the domain knowledge of manufacturing plants. In [145], ontology-based semantic modelling has been studied for systems of systems. They also combine semantic models and results from High-Level Architecture (HLA) based simulation to enable the cognitive capabilities of the CDT. Knowledge graph is one of the popular approaches to represent ontology [168]. In [116], they consider multiple virtual models across the entire system lifecycle. The authors use the knowledge graph as an enabler to represent the ontology for each model. However, ontologies only capture specifically semantic-related knowledge, which is usually domain knowledge traditionally managed by domain experts. Ontologies are not designed to model other types of knowledge, especially related to the self-awareness of the DT.

There have been some initial attempts to categorise the knowledge and its management involved in cognitive DTs. In [208], the authors define four types of knowledge for the cognition of the DT: definitional, deductive, inductive, and creative. They propose the notion of *Actionable Cognitive Twins (ACT)*, where the actionable capabilities utilise knowledge for decision-making. Specifically, the actionable dimension includes: "(1) the execution of a knowledge process (run a simulation, AI model, or execute some reasoning rules), (2) autonomously execute some actions in the manufacturing plant, and (3) provide decision-making options to the user." An architecture for the essential modules of an ACT is proposed, where the knowledge graph is a central module in reasoning, simulation-based and AI-based analysis, and decision-making. In [158], the authors apply the cognitive theory of *Learning Intelligent Distribution Agent (LIDA)* to DTs and propose an architecture for unmanned maintenance of machine tools. The cognitive cycle of LIDA has four phases: sensory, memory, attention, and execution. The capabilities of the LIDA cognitive-based

DT include self-construction, self-evaluation, and self-optimisation.

Despite the efforts, research is still in its infancy in designing fine-grained management and utilising knowledge for cognitive digital twins. In the LIDA cognitive-based DT mentioned above [79], it is the *memory*, a specific type of temporal knowledge, that is categorised based on the type of learned structures, function, and duration [79]. Existing publications focus on specific models and technologies such as ontology and knowledge graph [208]. Although the aspect of knowledge has been investigated in the literature [208, 158], the current research does not focus on the perspective of self-awareness, where knowledge is acquired, utilised, and updated dynamically at fine-grained levels. Self-awareness is essential as it captures the fundamental processes of runtime knowledge management to enable cognitive behaviours [143]. Therefore, there is a gap in delineating fine-grained knowledge management by addressing the self-awareness capability of the cognitive DT with no dependency on specific technologies. According to the proposed classification model in Figure 4.1, it can be argued that a cognitive digital twin should integrate self-adaption, prescriptive simulation what-if analysis, and online learning with human-like fine-grained knowledge management. Table 4.1 shows the classification of the existing works from the previously mentioned three dimensions of intelligence. Although [77, 204, 25] all reach the highest level along each dimension, they do not involve fine-grained runtime knowledge management. To approach this vision of augmenting DT with fine-grained runtime knowledge management and the highest level of intelligence in all three dimensions, this thesis proposes a reference model that integrates simulation and self-awareness of fine-grained knowledge in the next section.

Table 4.1: Classification on the intelligence of DT.

References	Autonomy		Analysis			Learning/Knowledge management		
	Human-in-the-loop/open-loop	Self-adaptive	Descriptive	Predictive	Prescriptive	N/A	Offline	Online
[10, 201]	✓		✓			✓		
[265, 152]	✓		✓					✓
[134, 67, 121, 122, 3, 223, 63]	✓				✓	✓		
[179, 252]	✓				✓			✓
[192, 28, 270, 52]		✓	✓			✓		
[225, 58, 120, 119]		✓	✓				✓	
[148, 271, 56]		✓	✓					✓
[207]		✓			✓	✓		
[64, 41, 245, 221, 133, 139]		✓			✓		✓	
[77, 204, 25]		✓			✓			✓
This thesis		✓			✓			✓

4.3 A Reference Model for Self-Aware Digital Twins

This thesis utilises knowledge in self-aware and self-adaptive systems to conceptualise a reference model for self-aware digital twins, intelligently working in coordination with the physical system. The proposed reference model is shown in Figure 4.2. This is inspired by the generic self-aware architecture provided in [49], which, in the case of the digital twin, is augmented with simulation capabilities. The reference model is a refinement for the *Cognitive Decision Maker* module in the generic architecture proposed in the previous chapter (Figure 3.4. The reference model takes a knowledge perspective and describes fine-grained multi-level knowledge management to achieve self-awareness.

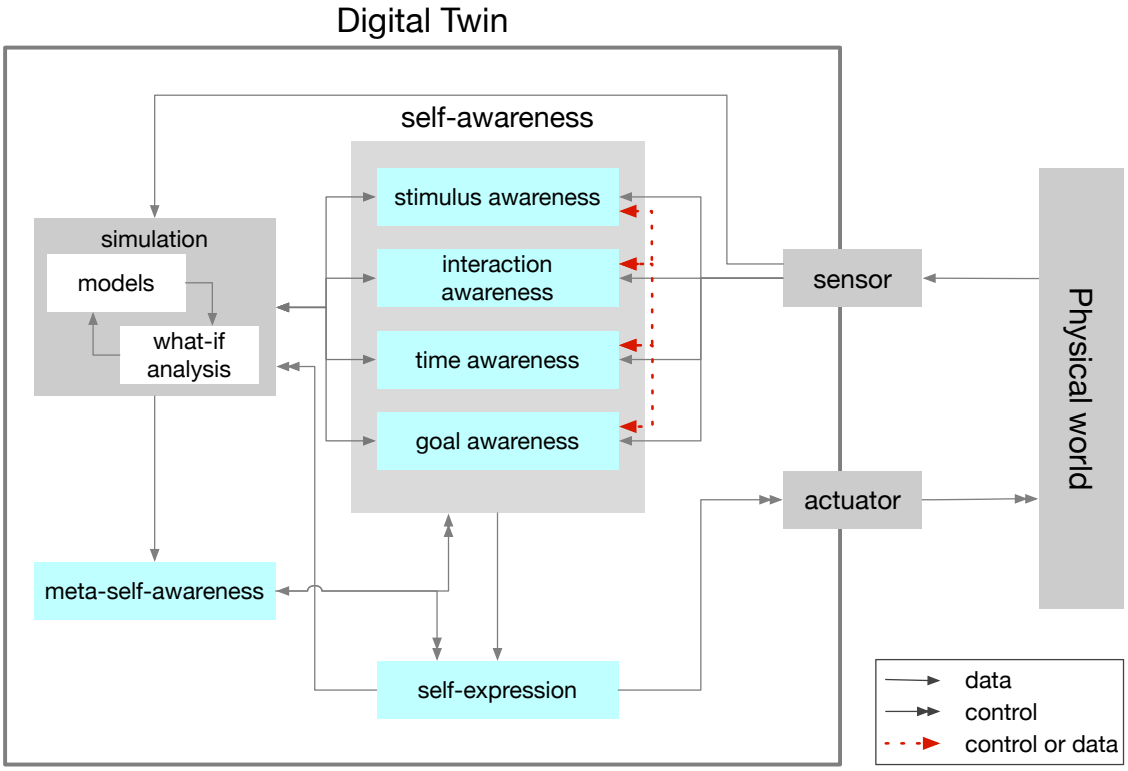


Figure 4.2: Reference model of a self-aware digital twin with full capabilities of self-awareness (a fully-cognitive digital twin).

The simulation contains models that reflect the behaviour of the physical world. What-if analysis is performed with the assistance of simulation models to generate new

knowledge. The self-awareness component is capable of reasoning and learning from both the physical world and the simulated world for the concerns of stimulus, interaction, goal and time. The meta-self-awareness analyses the performance history of both, the self-awareness and the simulation components. It acquires new knowledge by evaluating trade-offs and/or realisation of the other four levels of self-awareness. The self-expression component is the one that enacts adaptation and behavioural changes to the physical world and/or the digital twin itself through knowledge gained by self-awareness and meta-self-awareness.

The twin is fully aware of both symbiotic worlds, namely the physical world and its digital replica. The digital twin monitors the data from the physical world continuously through sensors. The data collected are then directed to the simulation component and the corresponding self-awareness component. The simulation component uses the data to update its virtual state in order to synchronise with the current states of the physical world. Four levels of self-awareness utilise real-time and historical data as well as the capability of simulation to analyse and acquire knowledge from the phenomenon monitored in the physical world, and from the predictive and prescriptive analysis of the simulated world. Then self-expression utilises the obtained knowledge to control the physical world through actuators.

Apart from the interaction with the physical world, there is also another internal cycle where meta-self-awareness utilises the performance history of self-awareness and the simulation history to acquire knowledge about the overall performance of the internal states of the digital twin. Then meta-self-awareness will initiate changes to the self-awareness component via self-expression to improve the performance of decision-making. Therefore, the self-aware digital twin is aware of knowledge obtained not only from the physical world, but also the replicated simulation in the digital world to enable the synergy between the two worlds.

4.4 An Illustrative Example

An example of city logistics is used to illustrate the mapping of the concept to the digital twin. As part of the mapping, this section will discuss different levels of knowledge that can enable intelligence and self-awareness.

Consider a city logistics scenario where emergency drone distribution for medical supplies is continuously dispensed from multiple warehouses across the city during pandemics (Figure 4.3). Parcels have dedicated destinations to be delivered, such as hospitals, care homes, and communities, and with hard time constraints for service delivery. A large number of smart drone carriers are deployed in the city for picking up and delivering parcels. Each drone has its dedicated depot where it should return to. However, as the supply and demand across the city are uncertain, the allocation of drones should be adjusted accordingly at each depot to meet the delivery demand and ensure a minimum number of drones are involved to lower the operation cost. Additionally, coordination between the smart drones is essential for meeting the service delivery goals, though the topology of this coordination can be highly dynamic, and drones have limited time for flying with varying durability in response to rain, wind, temperature, etc.

The smart drone is the task-related entity that is central to fulfilling the delivery tasks. Relevant data to be sensed can be its speed, position, current load (which parcels are onboard), load capacity, reliability over time in meeting its service delivery, remaining flying times, etc. Data of the parcel demand includes the location, destination, and delivery time window of each parcel. Other data outside of the logistics network are also needed, such as weather conditions.

With such a complex, uncertain, decentralised and highly dynamic network of flying drones, a versatile managing architecture assisted with an intelligent digital twin would be

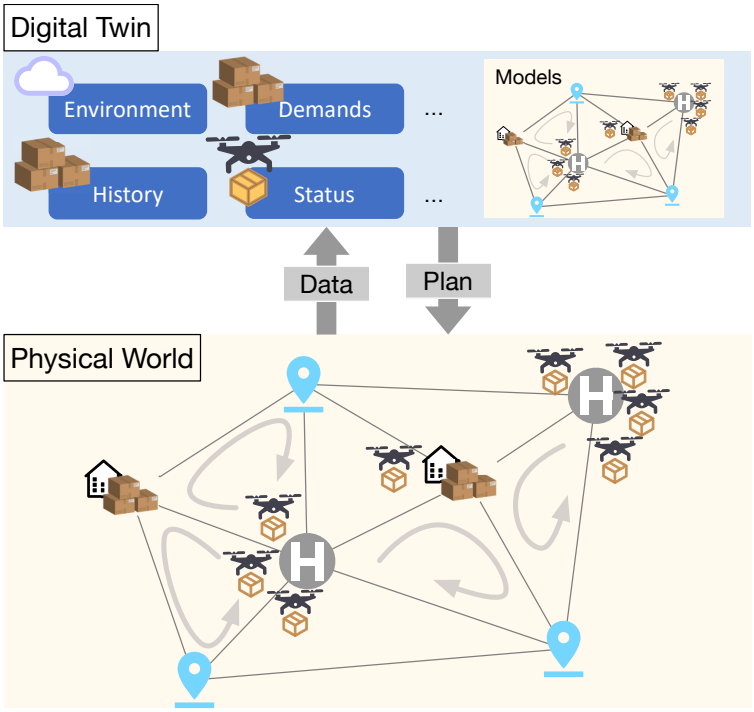


Figure 4.3: A city logistics example.

needed to leverage techniques involving optimisation, simulation, real-time data processing, analytics and learning for the purpose of intelligent decision making. The digital twin continuously monitors the data transmitted from sensors across the logistics network. When there is an unpredicted peak of delivery demands that is identified in real-time, the digital twin can assist the real-time physical network in evaluating different delivery strategies through a sequence of what-if simulation analysis, including the number and type of drones needed, their dependability to these conditions, where those drones should be dispatched. With the simulation result, actions can be taken in the real world to mitigate the delivery peak.

4.5 Mapping to Self-Awareness

This section outlines the different levels of self-awareness of the digital twin and how they would map to the motivating example.

4.5.1 Stimulus-Aware Digital Twin

A Stimulus-aware digital twin is the basis for all the other self-aware twin types. It is essentially reactive in nature; the twin can demonstrate how detecting and reacting to stimuli and events in the monitored environment can influence the state and behaviour of the observed system. Simulation and what-if analysis are not involved, because they both utilise time as the basis of predictive or prescriptive analysis for future outcomes. In our example, a stimulus-aware digital twin demonstrates how the system can react and adapt to events requiring routing optimisation with predefined policies and objectives. Examples of objectives can be to minimise the average delivery time of all parcels and increase delivery coverage. Examples of constraints may involve the time window requirement for the delivery of each parcel, environmental factors affecting delivery, drone flying time, etc.

4.5.2 Interaction-Aware Digital Twin

The capability of an interaction-aware digital twin is twofold. First, an interaction-aware digital twin operates on the knowledge acquired from the interaction of drones in the digital world – that interaction can relate to the historical or simulated eventuality of drones interacting in the physical world and its corresponding impact in meeting the network's overall objectives and constraints. Second, interaction-awareness can also be steered by data fed from the physical world on the interaction between drones; when such knowledge is processed by the digital world, it can enrich the knowledge of the digital world for more informed simulation, planning and decision-making that can be fed back to the real system.

4.5.3 Time-Aware Digital Twin

A time-aware digital twin utilises data related to the system's performance when autonomously responding to stimuli. This twin can support automatic or semi-automatic predictive and proactive planning informed by historical knowledge. For the motivating example, a time-aware digital twin can help in predictive capacity planning, such as increasing or decreasing the number of drones to cater for demand peak or off-peak, so that the pre-defined constraints in stimulus awareness are still met – parcels should be delivered within time windows. Time awareness can assist in proactive planning for the future (e.g. three hours later) delivery demands in advance. With this capability, the digital twin evaluates periodically if the current strategy will encounter performance degradation in the future, e.g. because of the shortage of drones in future delivery peaks. If true, new decisions will be generated and tested through what-if analysis and then enacted before the degradation occurs. Additionally, a time-aware digital twin can monitor time-variant feedback from the physical world to update the knowledge base in the digital world in an "info-symbiotic" data-driven style. Historical data related to the performance of drones in meeting their objectives and their routing strategies in the physical environment can be fed back into the digital twin to provide the basis of what-if analysis, simulation and planning – whether offline or online steered by time-aware knowledge.

4.5.4 Goal-Aware Digital Twin

A goal-aware digital twin has explicit knowledge about its goals and desirable states. The combination of design time and runtime goals facilitates the response to current phenomena measured from the physical world with goal-oriented planning. Goals can be discussed individually (e.g. each individual drone fulfils its own delivery tasks), or collectively (e.g. the whole delivery network copes with delivery peak). Goal awareness may also provide

insight into changing the runtime goals with the aid of simulation. During the what-if decision analysis, if no solution can meet the goals, the goals can be loosened to allow a suboptimal solution. In the motivating example, a desirable state of the physical world is one where most parcels can be delivered without constraint violation within a certain period of time. A state is undesirable if a large percentage of parcels are delivered within a longer than the one required, which might indicate a shortage of drones. The planning objective can be to control the physical world to avoid an undesirable state and/or to transit from an undesirable state to a desirable one.

4.5.5 Meta-Self-Aware Digital Twin

A meta-self-aware digital twin is aware of the other levels of self-awareness and adapts how they are realised, and/or decides the trade-off between the different levels of self-awareness. The full-capacity self-aware digital twin in Fig. 4 can illustrate meta-self-awareness. In the example, the digital twin can make predictions periodically with a certain frequency to detect possible future performance degradation. The meta-self-aware digital twin is able to adjust the frequency of such predictions to reduce overheads. A high frequency of prediction is not necessary when the delivery demands will remain stable in the near future, which may lead to high computational and monetary costs. Meta-self-awareness is able to minimise the frequency of prediction when the delivery demands are not and will not be changing dramatically. Another example can be to change the planning algorithm used in stimulus awareness. Due to the dynamic nature of the physical world, the performance of planning algorithms may vary with time and scenarios. The objective can be to select an algorithm among multiple candidates based on performance trade-offs.

4.6 Discussion

Different design patterns have been described in [49] for different levels of self-awareness that can be used to implement self-aware digital twins. Similarly, Table 4.2 provides an example of attributes that can enable or instantiate the different awareness levels.

The architecture in Figure 4.2 depicts a fully-cognitive twin; in the general case, not all different levels of self-awareness need to be present. For instance, time awareness is only valid when abundant historical data have been accumulated over time through the choices made by stimulus awareness in the past. Also, the trade-off between computational overhead, response time, and quality of adaptation among different levels of awareness remains a challenge for self-aware digital twins. Furthermore, on the levels of autonomy that the managed systems have without the aid of the digital twin, the focus of the self-awareness in a digital twin will vary: (i) If the managed system has its own reactive way of reasoning and acting, the digital twin will assist solely on the collective goals of the physical world. A challenge will be the coordination of knowledge gained from both the digital twin and the managed system. A control handover can happen when the digital twin has more insights than the managed system. (ii) If the managed system is controlled entirely by a digital twin, the digital twin will provide planning for not only collective goals, but also individual goals. In such a case, there is no knowledge coordination between the physical world and the digital twin.

Another challenge is the degree of distribution of a digital twin. As the case in the smart logistics scenario, a managed system can be a complex system consisting of multiple decentralised interacting components (e.g. the drones). A range of design options is available, from a centralised digital twin of the entire system (e.g. an agent-based model of the city and the drones) to a fully decentralised system where each component has its own twin with a limited view of the world, to a hierarchy of communicating digital twins with variable degrees of self-awareness.

Table 4.2: Attributes that instantiate self-awareness.

Attributes	**Awareness**				
	Stimulus-	***Interaction-***	***Time-***	***Goal-***	***Meta-self-***
Parameters	Stimuli, events and states (e.g., load, performance, battery life, etc.)	Source of data, topology of physical coordination (e.g. captured using object models, network analysis)	Historical data related to events, states, etc.	Individual or collective goals	Performance of other levels of awareness
Techniques	Conditions-actions rules, threshold-based algorithms	Coordination between nodes taking bids for tasks; knowledge sharing and negotiation	Statistical techniques, machine learning classifiers	Utility-based optimisation, reinforcement learning	Bucket of models, bandit solver
Purposes	Events monitoring; parameter updates; discrepancy mitigation	Coordinated decision making	Prediction, history-based adaptation	Maximising decision utility	Coordination between levels of awareness
Reactive or proactive	Reactive	Both	Both	Both	Both
Simulation and what-if analysis	N	Optional	Y	Optional	Optional

4.7 Related Work

There have already been attempts to develop self-aware digital twins, either explicitly or implicitly. However, none of the works explicitly consider knowledge in finer grain. They only implicitly discuss limited facets of knowledge. In [14], a systematic framework is provided for the evaluation of the intelligence of digital twins, but no specific manifestations are suggested.

In [224], the concept of a data-driven self-aware digital twin is discussed in the context of the 3D laser cutting process. The main purpose of self-awareness is to understand the present and predict future behaviour based on analysis of past data to support root-cause analysis. Apart from stimulus awareness, goal and time awareness are implicitly involved.

In [119], a DDDAS system is extended to a self-aware digital twin to support real-time path planning of an unmanned aerial vehicle (UAV) according to its structural integrity. In this work, stimulus awareness and goal awareness are implicitly involved.

In [206], a dynamic data-driven approach is applied to the digital twin model for 3D printer products. The digital twin is a machine learning model that predicts the surface texture and dimension of the product to be printed by using environmental parameters from sensors as input. Stimulus awareness and time awareness are implicitly involved.

In [207], a 3D simulation model is used as the mental model for the path planning of a mobile robot. The robot simulates all the possible future paths resulting from different initial parameters. Stimulus- and time-awareness are implicitly involved, the latter to predict future path trajectories.

Previous research has explored the integration of DDDAS or symbiotic simulation with self-awareness to facilitate intelligent decision-making in real-time, but no work has been applied to the concept of digital twins. In [127, 126, 125], a cognitive, self-adaptive

DDDAS agent is presented for the management of info-symbiotic agent-based simulation for social systems. In [69], a symbiotic simulator is introduced to act as the meta-self-awareness for volunteer computing. It simulates the functionalities of other levels of self-awareness by running what-if analysis to evaluate the outcomes of alternative awareness levels that the self-aware managing system could have chosen for a previous adaptation. Then a decision-maker component will decide if such alternative awareness levels will be enacted.

4.8 Summary

This chapter has outlined the landscape of self-aware digital twins and has also proposed a conceptual framework for self-aware digital twins focusing on fine-grained levels of knowledge. The framework leverages the combination of the conventional digital twin and the generic self-awareness architecture to facilitate cognitive intelligent planning and decision-making. Different levels of self-awareness and an explicit simulation component are central to the framework. The self-aware digital twin focuses on knowledge acquired not only from the physical world but also from the digital world. Further, this chapter has discussed how self-awareness can steer the synergy between the two worlds for more informed simulation and decision-making.

Chapter Five

Physical-to-Virtual: Knowledge Equivalence in Digital Twins

The previous chapter has introduced the concept of the self-aware DT and outlined an architecture for cognitive digital twins that can make informed decisions via predictive and prescriptive what-if analysis in the simulation environment, leveraging knowledge awareness and cognition capabilities. The main question examined is whether and how the DT can be utilised to enhance intelligence in the physical system. This chapter aspires to address a different challenge, namely how best to decide the equivalence between a cognitive digital twin and its self-aware physical counterpart. To address this challenge, this chapter focuses on the equivalence manager of the generic reference architecture presented in Chapter 3, and enhances it with equivalence-checking capabilities. The novel idea of knowledge equivalence is first presented and defined. In achieving equivalence, this chapter proposes knowledge equivalence checking as a cost-efficient approach to maintain the equivalence of the digital twin, by reducing unnecessary updates while still maintaining a valid model. This chapter relates to the P2V direction of mutual intelligence enrichment.

5.1 Overview

Predictions and analysis made by the digital twin can easily become obsolete when the physical world is highly dynamic and evolves in real-time, and when the DT's model drifts, decays and/or is not updated to reflect on changes in the physical world. The requirement for high fidelity of the digital twin system poses an important challenge to decide at which point a digital twin can indeed be considered a replica and not a mere abstract representation model of the physical system. In other words, fidelity replication begs the question: when is a model considered equivalent to the real system, and therefore, the model can be trusted, and no further updates are required? Too frequent updates in the model introduce excessive communication and computation overheads, while large delays in updating the model force it to drift from reality [214]. This chapter aims to address this question in the context of DTs of intelligent systems, from the perspective of knowledge.

For intelligent systems functioning in a dynamic environment, both the observed phenomenon of the physical environment and the internal knowledge bases of the intelligent systems evolve in real-time. It thus becomes essential to adopt a new research viewpoint that guarantees the DT also replicates the knowledge evolution of the intelligent systems equivalently. Knowledge synchronisation and equivalence become of paramount importance and prerequisite for meaningful, effective, and efficient control and analysis, as knowledge drift and decay can make the twining obsolete. Additionally, maintaining knowledge equivalence can subsume other fundamental issues that can be rooted in data sources, sensing, processing, actuating, control, etc. This can be particularly important in DTs, supporting intelligent systems that are grounded on knowledge, covering dimensions that relate to stimuli, time, interaction, goals, etc.

Current approaches for building Digital Twins do not provide explicit solutions to the problem of knowledge equivalence as a fundamental problem for ensuring fidelity between

the physical and digital worlds. This chapter specifically looks at knowledge equivalence in the design of intelligent systems, assisted by the DT. In particular, it investigates the challenge of real-time modelling of autonomous systems which can accumulate knowledge through the DT to support intelligence and how knowledge equivalence can be engineered and/or reasoned about in this setting.

The work presented in this chapter assumes that a digital twin is an imperfect artefact when attempting to "follow" the physical system, and calls for knowledge comparison between the digital and physical worlds [84]. The agents operating in the physical world are considered **interaction-aware**, that is, being able to learn models capturing knowledge specifically about their interaction with other agents and the environment. Each agent maintains its own knowledge base and uses the interaction model to plan its actions. The knowledge of interaction is modelled as a weighted graph by each agent.

The objective of this chapter is (i) to inform when and how often to update the DT knowledge, and (ii) what is an acceptable tolerance and knowledge deviation level between the digital and physical worlds. The chapter posits that utilising knowledge equivalence as opposed to *state equivalence* can alleviate the overheads while improving simulation validity and can be a more efficient and effective approach for the DT of intelligent systems. The contribution of this chapter is as follows:

- The formulation of the problem of knowledge equivalence for the DT of an intelligent physical system. This includes an outline of the threats to knowledge equivalence and their detection and impact, as well as a set of metrics that measure the equivalence of fine-grained knowledge related to interaction.
- The refinement of the holistic architecture that specifically focuses on knowledge equivalence.

- An online knowledge equivalence checking framework that identifies discrepancies and determines when to update the DT.
- Demonstration and quantitative evaluation of the proposed framework based on a smart camera network scenario where agents are interaction-aware.

5.2 Related Work: Equivalence in Digital Twins

To support accurate simulations by DTs, models that are equivalent to the physical world are needed. This chapter focuses on the DT modelling of intelligent systems. The next subsection delineates the scope of intelligent systems in this chapter.

The concept of equivalence has been investigated in different domains and problems including model verification and validation [215, 43, 234], dynamical systems [217], model checking [32, 54], program equivalence [220, 53], software evolution [250], knowledge representation [92], bisimulation of concurrent systems [61], agent-based model adaptation [95], VLSI design[36, 275], and industrial control systems [241]. Grounding models to data in an online feedback loop has been extensively investigated in the context of DDDAS systems (as mentioned earlier in Chapter 2), a comprehensive review of approaches to implement the feedback loop is provided in [34].

The rest of the section provides an overview of approaches and techniques that focus specifically on the issue of twinning addressing the problems of how to decide when the model needs to be updated and how the model is updated. Table 5.1 summarises these approaches. Three main classes of approaches are distinguished, namely replicating the sensor data without any comparison; explicitly identifying discrepancies between the model and the real system; continuous model calibration. With regard to model adaptation, four elements that may be updated are state variables, parameters, behavioural rules and input.

Table 5.1: Related work for online equivalence.

Category	Reference	Comparison entities	Update	Method
Replication	[10, 134, 55]	-	State	
Replication	[68]	-	Input (stimuli)	Stimuli identification
Replication	[118]	State (with the previous synchronised value)	State and output	Threshold-based approximate synchronisation
Equivalence Checking	[126]	Knowledge (rules of agents)	-	Association rule mining
Equivalence Checking	[84]	Outputs given the same inputs	-	Gaussian Mixture Model & Hidden Markov Model
Equivalence Checking	[97]	State	Parameter	Multi-agent reinforcement learning
Equivalence Checking	[176]	State	Parameter	Reinforcement learning
Equivalence Checking	[155]	Output sequence	Parameter	Online validation by time series analysis
Equivalence Checking	[80]	Time	N/A	Time discrepancy checking
Online Calibration	[190, 96]	State	Parameter	Optimisation
Online Calibration	[277]	Output	Input	Optimisation
Online Calibration	[99]	Statistical test on states	Parameter	Optimisation
Online Calibration	[120]	N/A	State	Bayesian inference
Online Calibration	[4]	N/A	N/A	Kalman filter or PID
Online Calibration	[115]	Output	Parameter	Error feedback
Online Calibration	[179]	State	Parameter	Online training
Online Calibration	[229]	N/A	State and parameter	Stochastic optimal control

5.2.1 Sensor Data Replication

This approach aims at achieving equivalence by directly replicating sensor data into the model. The sensor data may be used to update the state of the model either at a fixed frequency [10] or dependent on the availability of data or event [134, 55]. The aim is to ensure the state of the model reflects that of the real system. The work in [228] studied adaptive synchronisation for balancing the trade-off between simulation accuracy and the costs associated with frequent updates. The problem is formulated as finding an optimal control policy that decides when to synchronise the state of the DT. However, their work assumes the DT has an accurate model of the physical system, which hinders the applicability to online decision-making under a dynamic environment.

In industrial control systems, a state usually refers to a hidden property of the system that cannot be obtained through sensors. Sensors can only measure the input and the resulting response (output) of the system in order to estimate the current state. In [68] the state is replicated by identifying and updating the stimuli inputs in industrial control systems. This work assumes the physical system and the twin system run the same program defined by a finite state machine. However, the state transition of the program is fixed and does not evolve over time, indicating no learning and awareness of knowledge are involved.

Kalasapura, Li, Liu, Chen, Wang, Abdelzaher, Caesar, Bhattacharyya, Kim, Wang, Kimberly, Eckhardt, and Osipychev [118] proposed a synchronisation protocol called Twin-Sync for bandwidth-limited IoT applications. The state can be adaptively synchronised, but whether to synchronise is decided by the real-world system, not the DT. They incorporate operator intent as high-level objectives and output of interests to guide the update. A shadow twin on each node decides whether to transmit new data of the state to the digital twin. Such a decision is made based on the difference between the current system state and the previous synchronised value of the state. If the difference exceeds a threshold determined

by the digital twin, then the shadow twin on the node will send the new state.

The work of this chapter instead focuses on twinning a system that maintains and evolves its own knowledge base. State replication alone is not enough to allow the DT to "mirror" all the perspectives and dynamics of the system. Real-time replication of the knowledge base is also needed for intelligent systems.

5.2.2 Discrepancy Checking

This approach focuses on explicitly comparing the model and the real system to identify discrepancies and decide whether and when to update the model. The AIMSS system has investigated this problem in the context of info-symbiotic (DDDAS) agent-based social simulations[127, 125, 126]. The system utilises an SAT solver to detect inconsistencies of different rule sets from the model and the real world dynamically identified via associative rule mining. The rules can be viewed as knowledge of the agents, however, this is performed offline. The discrepancy detector in [84] utilises a Gaussian mixture model to compare the outputs of the real system and model given the same inputs and detect anomalies due to the missing mode of the model. Checking the inconsistency of states using reinforcement learning agents is described in [97]. The agents take the state difference as inputs and decide whether to update and which parameter to calibrate. In addition, the authors in [176] also apply reinforcement learning to update the parameters (velocity) in robot applications, by using position differences as rewords. In [155], the authors propose an online validation method for the DT of manufacturing systems. Dynamic Time Warping, which is a time series analysis algorithm, is used to compare the inter-departure time sequences of the manufacturing queue produced by the physical system and a Discrete Event Simulation model. A threshold is used to alert the validity of the model. If not valid, the parameters for the distribution of processing time are re-fitted with the recently acquired data.

Additionally, the problem of time discrepancy for co-simulation-based DTs has been addressed in [80]. The digital twin and its physical twin may be out-of-sync due to network issues such as temporary network degradation or network drop. These network issues may cause delays in data transmission between the physical twin and the digital twin. The authors have designed mechanisms that can identify time discrepancies. If the discrepancy exceeds a predefined threshold, then the digital twin will degrade to a digital shadow that only receives data from the physical twin but does not send control messages to the physical twin anymore. When the synchronisation is established, The digital shadow will be recovered back to be a digital twin.

Existing effort in checking the equivalence between the DT and the real system is limited to the comparison of states, input or output. There is still a lack of attention to knowledge equivalence checking and comparison in order to identify whether the DT maintains a virtual intelligent agent that possesses knowledge equivalent to the knowledge of the real-world intelligent agent. As explained in section 5.1, this is particularly important for intelligent systems.

5.2.3 Continuous Online Calibration

Continuous online model calibration is also utilised for maintaining an equivalent model. Calibration focuses on continuously tuning the model parameters, typically at a fixed frequency. Approaches in this category include the use of: difference/error between observed and predicted values of the state variables as the objective function to optimise the selection of parameters or inputs [190, 96, 277]; statistical tests on simulated and real traffic flows as calibration criteria [99]; Bayesian inference in the context of unmanned aerial vehicle (UAV) applications [120]; Kalman filter [4]; the error between observed output and predicted output as feedback for the calibration of clock models [115]; and online calibration by training deep

learning models with streaming data (state variables) [179]. In addition, Tan and Matta [229] formulate the optimal digital twin synchronisation problem as a stochastic control problem, in which the objective is to find an optimal policy that determines when to update the predictions and model parameters. Specifically, their approach enables deciding whether to update the performance predictions with the latest observation, whether to update the model parameters, and whether to increase the detail level of the simulation model. The approach also minimises prediction bias and synchronisation overhead.

5.3 Knowledge Equivalence

Ensuring equivalence between the DT and the real world is essential for accurate simulation predictions and analysis. When "twinning" a system that learns knowledge, an additional requirement should be enforced on maintaining equivalence between the knowledge base of the physical agent and that of the corresponding virtual agent. This section defines knowledge equivalence and presents different situations that lead to the knowledge deviation between the virtual system and the physical system. The work presented in this chapter focuses specifically on interaction-awareness. Though this awareness level deals with interactions, it backs itself up with both stimuli- and time-awareness knowledge in relation to these interactions. Knowledge can be implemented using different data structures. In this chapter, the knowledge of an agent is represented as a weighted graph to model interaction with other agents.

5.3.1 Defining Knowledge Equivalence

First, consider a pair of physical and virtual agents with knowledge k_i and k_j respectively. The pair has identical $\{A, P, K, \phi, \pi\}$ as modelled in Section 3.2.1.1. If the two agents are

said to exhibit equivalence in their knowledge, then given the same perception, they should output and express the same action. That is, the two agents are deemed to be knowledge equivalent if and only if:

$$\forall p \in P,\ \pi(p, k_i) = \pi(p, k_j) \tag{5.1}$$

Notice that as mentioned in Section 3.2.1.1, the output of π can either be one single action or a probability distribution of multiple actions. The comparison between two probability distributions can be quantified by well-established statistical techniques (e.g. histograms [70]). Such a definition implicitly implies the tolerance of discrepancies: if and only if the two pieces of knowledge lead to the same response given the same input, they are regarded as equivalent, no matter how different the values of the two are.

For a multi-agent system and its model, the pair of agents of the virtual and physical worlds should strictly exhibit knowledge equivalent, if: for every agent α_i and its model α_i', equation (5.1) should hold.

However, it can become difficult to identify whether a multi-agent system and its digital twin model are equivalent, because exhaustively evaluating all possible perceptions is infeasible when the state space is too large or when the number of agents is also large. Additionally, it is equally time-consuming to compare all possible perceptions at every wall clock time tick.

5.3.2 Threats to Knowledge Equivalence

Assuming a correct and validated model by design, equivalence between the DT and the physical world can be threatened due to various reasons, thus causing deviation between the knowledge or/and states of the two intelligent systems. Some of the major threats to equivalence are listed below.

5.3.2.1 Temporal

Temporal deviation refers to changes in the physical environment not being timely reflected in the DT model. This type of deviation can be attributed to the inadequate or suboptimal frequency of timely updates. The deviation is often noticed, when the time interval between two updates is long.

The updates can be infrequent and not timely for several reasons. Firstly, hardware bottleneck. The data processing capacity of the DT or network bandwidth may not, for example, sustain a high volume of incoming sensor data in a short period of time. Another reason is the need to minimise the energy consumption of the sensors. Frequent sensing and data transmission to the DT will cause the sensor may consume excessive energy, limiting its dependability and usage lifetime in the long run. Thirdly, a sensor can be temporarily put into sleep mode when its data can be inferred by other sensors [173], but at the risk of inaccurate estimation. Fourthly, the user (human or application) of the model may not require fine-grained high-frequency timely information. Finally, sensors may fail to transmit data due to hardware failure.

The consequence of temporal deviation is that at any time step between two consecutive updates, the status of the physical world can be unknown to the DT. The DT may rely on data updates or estimates of previous runs, or may extrapolate its knowledge from other data sources. To detect such a deviation, comparisons are needed at each updating time step.

5.3.2.2 Unpredictable Environment Changes

Deviation in knowledge can also happen due to unpredictable environmental changes that a running physical system may experience, and guards for triggering knowledge updates for

these changes and surprises are missing in the design. One typical cause of unpredictability is partial observability – the detail of the entire environment is not fully observed through sensors. Another cause is the model being designed with biased or simplified assumptions. For example, the physical environment can be highly dynamic such that the environment model may not fully model all possible unforeseen scenarios, especially emergency events like car accidents in traffic-related applications, the appearance of sudden objects obstructing the cars etc. Since agents interact with the environment, the deviation between the virtual and physical environment can cause the knowledge of virtual agents and physical agents to diverge, leading to different behaviours. Such a deviation will invalidate knowledge accumulated by the virtual system. To detect this type of deviation, continuous validation can be applied.

5.3.2.3 Nondeterministic Agent Behaviours

Intelligent self-aware agent may not act deterministically. If the physical agent and its virtual agent replica both act in a probabilistic manner, they are very likely to choose different actions even given the same perception and knowledge base. Different actions lead to different feedback from the environment. The two agents then may revise their knowledge bases differently given the different feedback. Different actions may also cause the virtual environment and physical environment to evolve in different trajectories. This type of deviation can be detected by directly comparing the actions made by the virtual agent and physical agent.

5.3.2.4 Model Drift

Similar to the discussion in [72], model drift refers to changes made in the DT not being reflected in the real world and vice versa. Note that model drift is not the same as the temporal

deviation. The DT may modify the virtual system and synchronise the same changes in the physical system, but the synchronisation may fail, causing the two systems to drift apart. To give an example, the DT instructs a virtual robot to carry a parcel to a new location in the simulation, and then intends to synchronise the same action to the physical robot. However, during the operation of the physical robot, the parcel drops halfway during transportation. Without additional verification, the DT believes the parcel in the physical world has been transported successfully to the desired destination. The aggregation of this type of deviation over time will cause the model to drastically deviate from the physical world. To detect this deviation, data from various additional sources are needed to verify the synchronisation. In the above example, a camera can be installed on the robot to monitor the condition of the parcel as a means of verification.

Anticipation of the above threats can be incorporated into the design of knowledge equivalence methods in order to mitigate possible deviation during runtime. In addition, these threats can be valid assumptions for the controlled experimental evaluation of the DT system prototypes. This chapter also uses combinations of threats to evaluate the knowledge equivalence-checking method proposed in Section 5.5.

5.4 A Refined Reference Architecture for Equivalence Management

This section provides refinement of the equivalence maintenance mechanisms in the proposed holistic architecture. Specifically, the internal components of the Equivalence Manager are designed with the assumption that the physical agents have levelled self-awareness capabilities. The decentralisation pattern of the Equivalence Manager is also proposed later in this

section.

5.4.1 Requirements and Assumptions

The equivalence manager compares the model to the physical world on a continuous basis. If there are any discrepancies, it will update the variables that characterise the current representation of the virtual world or adapt/calibrate the model accordingly. It is assumed that data regarding the state of the physical environment and knowledge of each agent are available to the DT through sensors. The Equivalence Manager should continuously compare the state and knowledge between the physical world and the virtual world. It checks whether the state and knowledge in the virtual world simulated by the model are equivalent to the physical world. If not equivalent, then the state and/or knowledge needs to be updated. The Equivalence Manager should also identify whether such inequivalence is caused by the inaccuracy or bias of the internal parameters of the model, thus informing model calibration.

The physical agents are assumed to be instantiated with the self-awareness paradigm. As introduced in section 2.1.2 and shown in Fig. 3.2, a self-aware agent can maintain knowledge at different levels. The self-awareness component constructs and modifies the knowledge base by its capabilities at one or multiple level(s) with sensor perceptions. The *self-expression* component expresses decisions and yields consequent actions entailed by the awareness level(s).

5.4.2 Equivalence Manager

To design the architecture for the above mechanism, this thesis refines the Knowledge Manager originally proposed in Figure 3.4. Since knowledge is divided into different levels in the agent's knowledge base, the equivalence manager should also maintain the equivalence of

each dimension of fine-grained knowledge. Fig. 5.1 shows the instantiation and refinement for the model equivalence of the holistic reference model when the intelligence of the agents is modelled by self-awareness.

The Equivalence Manager is composed of an equivalence checker and an updater. The equivalence checker (EC) contains a state equivalence checker, a knowledge equivalence checker, and an aggregator. The state EC checks equivalence by comparing the state transition of the physical world and the model. The knowledge EC is further refined into four levels according to the levels of self-awareness: stimulus EC, interaction EC, time EC, and goal EC. Each fine-grained knowledge checker is responsible for the knowledge comparison of that level. For instance, the interaction EC monitors the interaction-related knowledge in the knowledge base of the physical agents and the virtual agents. When the interaction-related knowledge in the physical and virtual agents deviate from each other, the equivalence manager should update the interaction knowledge base of each virtual agent. The aggregator coordinates the different ECs to decide which EC(s) to utilise. Such coordination may require a more sophisticated design and is out of the scope of this thesis. Finally, inequivalence detected by ECs will trigger the updater to update the model by simulation re-initialisation (on the states and/or knowledge) and/or model adaptation.

5.4.2.1 Decentralisation of Equivalence Manager

The equivalence manager can be implemented as a central entity or decentralised to each virtual agent. In a centralised manner as shown in Fig. 5.1, the manager checks the knowledge of all agents, and updates them at the same time. For decentralisation, each virtual agent is assigned an equivalence manager as shown in Fig. 5.2. Each equivalence manager only checks the knowledge and state of the agent that it is attached to. All equivalence managers also work asynchronously such that each manager decides by itself when to update its

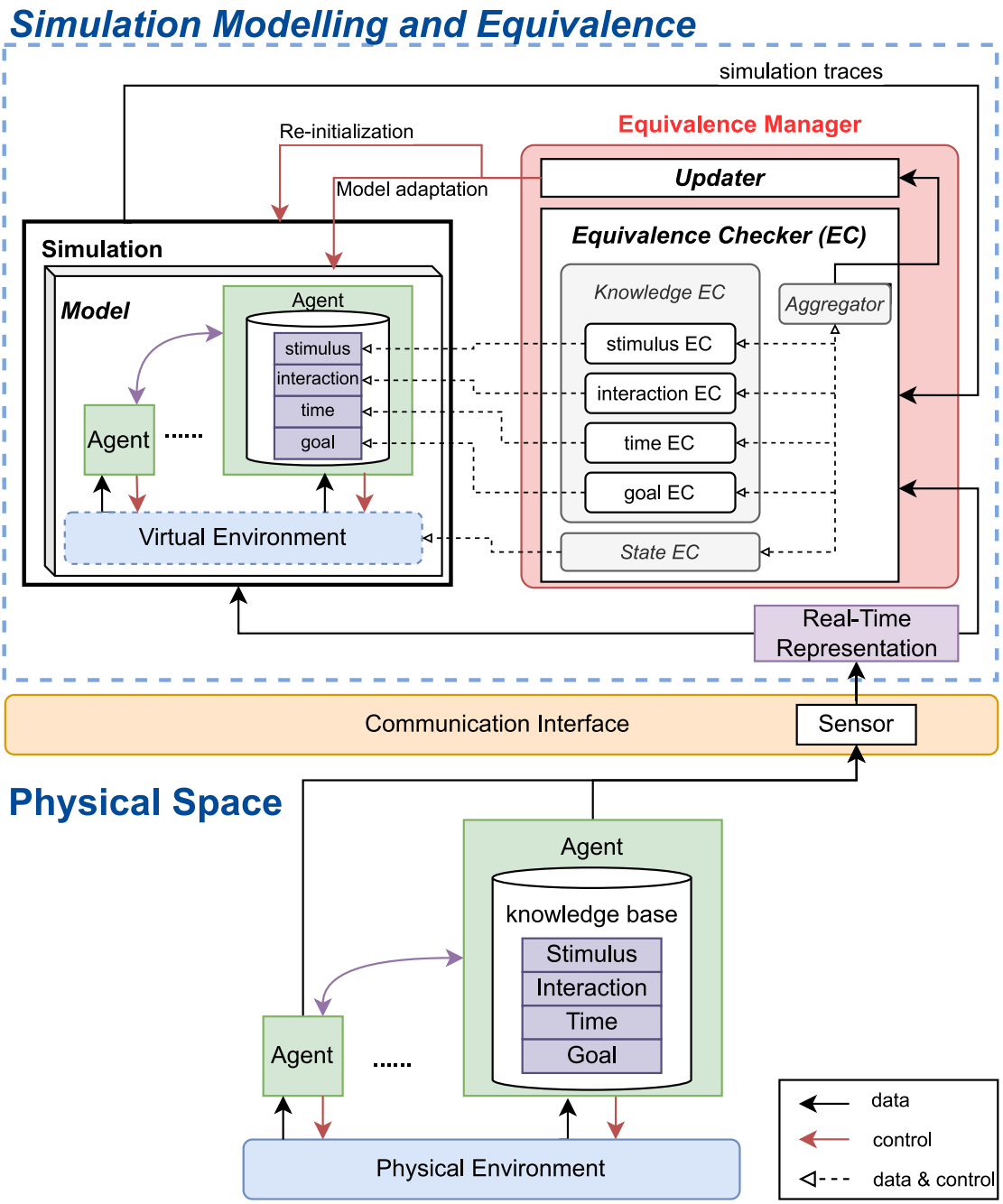


Figure 5.1: Instantiation of Fig. 3.4 with Self-Awareness.

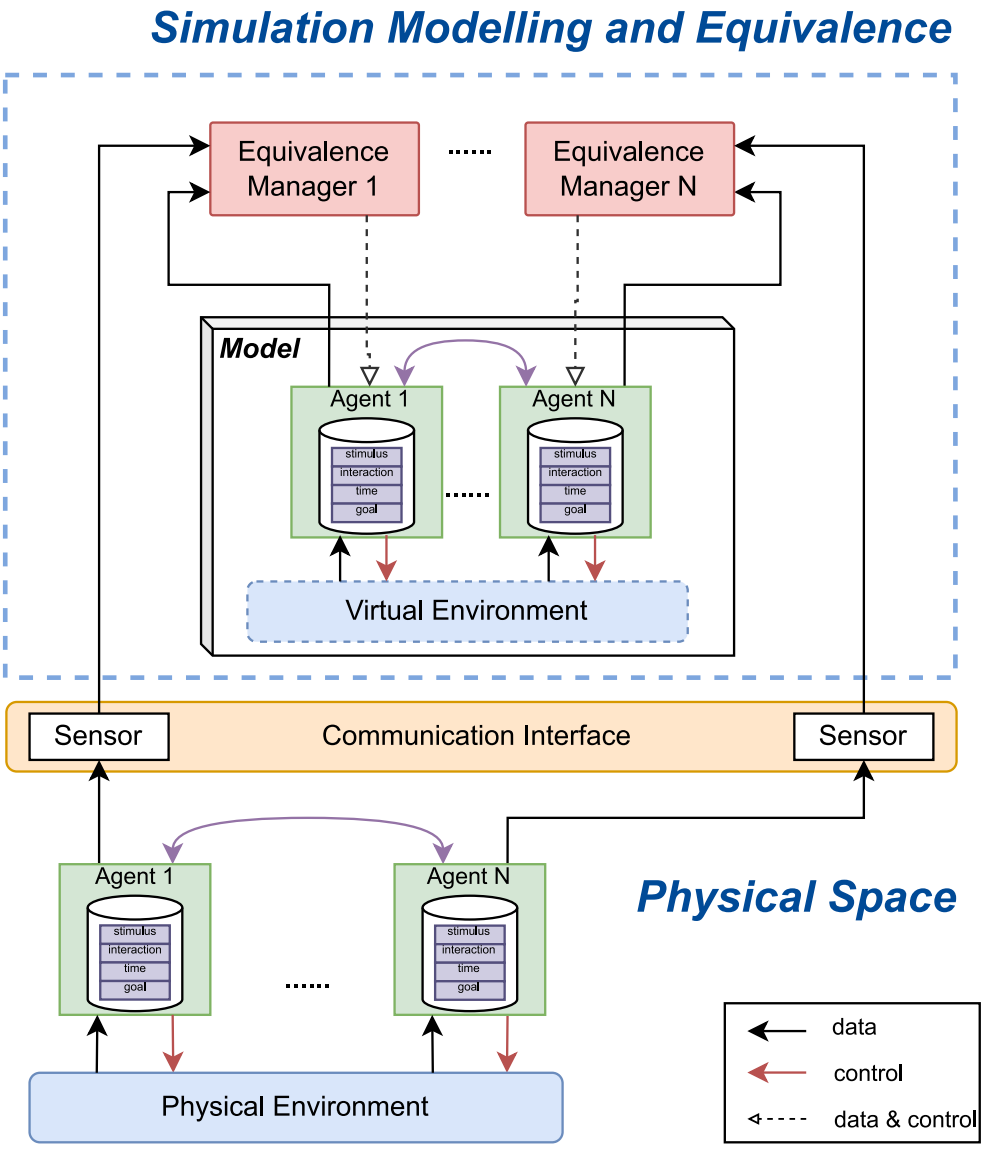


Figure 5.2: Decentralisation of Equivalence Managers.

virtual agent. This thesis particularly focuses on a centralised equivalence manager and the knowledge related to interaction, i.e. equivalence checking by the interaction equivalence checker. The decentralisation is left as a future work.

5.5 A Methodology for Knowledge Equivalence Checking

The key to knowledge equivalence checking lies in the identification of potential knowledge drift through continuous online comparison. In this regard, two fundamental questions arise: how to quantitatively identify knowledge drift, and when to update the simulation model. The type of available sensor data dictates the possible continuous comparison approaches for online equivalence checking. Fig. 3.3 shows the model used in this chapter and it involves three types of available data that can be used in comparison: environment state, knowledge of agents, and actions of agents. While the comparison of the state is a straightforward approach, which is also reviewed in section 5.2, it does not consider the fact that physical agents maintain self-evolving knowledge bases. Therefore, two novel ways of comparison are proposed in this section based on the rest two types of data to directly check the equivalence of knowledge: knowledge comparison and action comparison. A threshold-based tolerance approach is adopted to trigger model updates.

5.5.1 Benefits of Knowledge Equivalence Checking

Traditionally, equivalence checking is largely based on state synchronisation and comparison between sensory data of the physical system and the output of the model or through the exhaustive exploration of transitions of states (see section 5.2). Such *state equivalence* checking

methods can suffer from heavy computational overheads, making them difficult to scale to large-scale settings, such as in DT-related studies where equivalence should be maintained in real-time or near real-time. This is exacerbated in highly dynamic applications where model adaptation is expected to be performed frequently to capture the up-to-date dynamics of the fast-evolving environment.

In systems encompassing intelligent agents, the dynamic non-deterministic behaviour of agents may also change the dynamics of the physical world and the world model. In such cases, the knowledge of the agents may become the determining factor affecting the deviation of these two respective components of the twin system, as it is the knowledge of the agents that determine their decisions, their actions and their effect on their environment. Using knowledge as a metric for equivalence checking such intelligent systems emerges therefore as a viable approach that could overcome the high overheads involved in using states since knowledge is a task-related abstraction of state.

5.5.2 Knowledge Comparison Method

Knowledge comparison refers to directly quantifying the similarity of knowledge possessed by the virtual agent and physical agent. The design of the similarity metric is specific to the type and representation of knowledge. Therefore, this chapter specifically focuses on agents that are interaction-aware.

5.5.2.1 Knowledge Representation and Drift

For systems where agents interact and collaborate with each other, a common practice is to represent interaction awareness as undirected weighted graphs [70, 75]. This chapter also adopts graphs to represent interaction-related knowledge. In the graph, each vertex

represents an agent, and the edge weights are the knowledge of the interaction between agents. An example is to use weights to represent the collaboration history and tendency between two agents, which can be modelled by artificial pheromones [75]. For any agent i, if it has a high weight value with a certain agent j, then the weight means agent i has been collaborating with j recently in the past, and i tends to collaborate more with j in the future, rather than other agents with lower weight values.

Globally for the entire system with m agents, there are at maximum $l = \binom{m}{2} = m \times (m-1)/2$ edges. Here w_j is used to denote the weight value of the jth edge. The weights of all l edges between physical agents at time t are represented as a vector:

$$\mathbf{w}(t) = (w_1(t), w_2(t), w_3(t), ..., w_l(t)) \tag{5.2}$$

Similarly, the edge weights between virtual agents at the current time step t are:

$$\mathbf{w}'(t) = (w'_1(t), w'_2(t), w'_3(t), ..., w'_l(t)) \tag{5.3}$$

Then to measure the knowledge drift of interaction-awareness between the simulation and physical world at time step t, this chapter defines a metric $drift()$ as the Euclidean distance between $\mathbf{w}(t)$ and $\mathbf{w}'(t)$:

$$drift(\mathbf{w}(t), \mathbf{w}'(t)) = |\mathbf{w}(t) - \mathbf{w}'(t)| = \sqrt{\sum_{i=1}^{l} (w_i(t) - w'_i(t))^2} \tag{5.4}$$

5.5.2.2 Workflow

A threshold-based comparison workflow is designed to identify inequivalence of knowledge, which is shown in Algorithm 1. It is assumed that the simulation proceeds in real-time at

the same pace as the wall clock time[1]. The algorithm continuously senses the knowledge bases in the simulated agents and physical agents at every wall clock time tick (lines 5-8). The time duration between two time ticks is specific to the application scenarios, and is out of the scope of this algorithm. The knowledge comparison is run every q time ticks. A time window of length l is used to calculate the cumulative knowledge drift within this window (lines 11-13), where the function $drift()$ is defined by Equation (5.4). Then if the drift value is larger than a threshold θ, the simulation is regarded as knowledge inequivalent to the physical world. The simulation then needs to be re-initialised with the latest world state and knowledge (lines 15, 16) as:

$$\sigma'_t = \sigma_t,\ where\ \forall i\ \sigma'_{i,t} = \sigma_{i,t} \tag{5.5}$$

$$\mathbf{w}'(t) = \mathbf{w}(t),\ where\ \forall i\ w'_i(t) = w_i(t) \tag{5.6}$$

5.5.3 Action Comparison Method

Algorithm 1 is only applicable when the drift of knowledge is clearly defined. However, for more generalised situations where knowledge may not be represented as a graph or cannot be easily quantified, different approaches are needed. This chapter proposes an alternative approach that compares actions made by virtual and physical agents, since the action is the direct result of knowledge, and varies based on the level of knowledge the agent has.

[1]The idea is to allow the simulation to run in parallel with the real world, such that at any moment, the simulation model always represents a snapshot of the real world, being neither ahead nor behind in time. Users would be able to inspect the current (estimated) status of the real world directly from the model, without waiting for the sensor data to be transmitted and collected. More importantly, such a real-time model is always ready to give predictions about how the real-world system would evolve starting from the current state of the world.

Algorithm 1: Knowledge Comparison.

Input: comparison time interval q, threshold θ, comparison time window l
Data: History of knowledge of all physical agents $\mathbf{K}$, history of knowledge of all virtual agents $\mathbf{K}'$, time t

$\mathbf{K} \leftarrow []$;
$\mathbf{K}' \leftarrow []$;
$t \leftarrow 0$;
while *True* **do**
  $\mathbf{w}'(t) \leftarrow$ sense knowledge from simulation;
  $\mathbf{w}(t) \leftarrow$ sense knowledge from real world;
  $\mathbf{K}'[t] \leftarrow \mathbf{w}'(t)$;
  $\mathbf{K}[t] \leftarrow \mathbf{w}(t)$;
  // Compare every q time ticks
  if t *mod* $q == 0$ **then**
    $d \leftarrow 0$;
    for j = *max(0, t-l) ... t* **do**
      $d \leftarrow d + drift(\mathbf{K}[j], \mathbf{K}'[j])$;
    end
    if $d > \theta$ **then**
      $\sigma_t \leftarrow$ physical world state at t ;
      Re-initialise simulation with σ_t, $\mathbf{K}[t]$;
    end
  end
  wait for the next wall clock time tick;
  $t \leftarrow t + 1$;
end

5.5.3.1 Action Deviation

In order to compare action, a metric that quantifies action deviation is proposed as follows. The main idea is to measure how the actions made by all the virtual agents are different from the physical agents.

First of all, all the agents are assumed to have the same action set A which contains a finite number of elements. Then, if the physical world is composed of m agents in total, the deviation is calculated as

Algorithm 2: Action Comparison.

Input: Comparison time interval q, threshold θ, comparison time window l
Data: History of actions by all physical agents $\mathbf{A}$, history of actions by all virtual agents $\mathbf{A}'$, time t

```
1  A ← [];
2  A' ← [];
3  t ← 0;
4  while True do
5      a_t ← sense actions from simulation;
6      a'_t ← sense actions from real world;
7      A[t] ← a_t ;
8      A'[t] ← a'_t ;
       // Compare every q time ticks
9      if t mod q == 0 then
10         d ← 0 ;
11         for j = max(0, t-l) ... t do
12             d ← d + deviation(A'[j], A[j]) ;
13         end
14         if d > θ then
15             k_t ← sense knowledge of all physical agents;
16             σ_t ← sense physical world state ;
17             Re-initialise simulation with σ_t, k_t ;
18         end
19     end
20     wait for the next wall clock time tick;
21     t ← t + 1;
22 end
```

$$deviation(\mathbf{a}', \mathbf{a}) = \frac{1}{m}\sum_{i}^{m} \mathbb{1}_{a'_i=a_i} \quad , where\ \mathbb{1}_{a'_i=a_i} = \begin{cases} 1 & a'_i = a_i \\ 0 & a'_i \neq a_i \end{cases} \tag{5.7}$$

where a_i and a'_i are the actions made by physical agent i and its virtual counterpart a'_i.

5.5.3.2 Workflow

Algorithm 2 illustrates the workflow of action comparison, which is similar to knowledge comparison. The main difference is in lines 5-8 and line 12, where the actions made by the

physical agents and virtual agents are recorded and compared. In addition, instead of using **w** to denote knowledge specifically represented as a graph, the symbol **k** is used to represent knowledge in order to be generic.

5.6 Evaluation

This section evaluates the effectiveness of the previously proposed two knowledge equivalence checking methods in maintaining an equivalent DT. The evaluation is conducted through a case study using smart mobile cameras. Maintaining knowledge equivalence involves two repetitive steps: checking for knowledge discrepancies and updating the knowledge. As mentioned in sections 5.3 and 5.5, the main premise of the chapter is that 1) maintaining knowledge equivalence is crucial for maintaining an equivalent DT model that replicates networked intelligent self-aware systems, and that 2) the methods used for checking knowledge equivalence are efficient and have low overhead in keeping the DT model up-to-date. Two factors are considered in the evaluation: 1) updating the model is resource-intensive, since it requires re-initialisation of all variables, which is time-consuming, and 2) comparison takes time and memory space.

The evaluation will utilise the following metrics:

- *Number of updates*: this metric corresponds to the total number of updates within a given time period. This metric is indicative of potential overheads that can be incurred as a result of the updates, where more updates may require frequent re-configuration and re-initialisation of the simulation. Once knowledge in-equivalence is identified, an update will follow; henceforth, the number of updates is equal to that of the number of in-equivalence checks.

- *Average utility deviation*: This metric relates to the difference between the task goal satisfaction (defined by a utility function) observed in the real system and the simulated modelled system. This metric indicates simulation validity, averaged over the time of observations. The utility function is application-specific, but the metric of utility deviation within n consecutive time units can be defined as:

$$\text{Avg. utility deviation} = \frac{1}{n}\sum_{t=1}^{n}|u(t) - u'(t)| \tag{5.8}$$

  where $u(t)$ is the utility of the real system at time t and $u'(t)$ is the utility of the modelled system at time t.

The minimisation of both metrics is essentially conflicting in objectives. When the model is updated more frequently, the simulation of the modelled system strives for fidelity with the real system. Conversely, with less frequent updates, the simulation can easily drift away from the real system. To address this problem, this chapter assesses each of the proposed checking methods by the *Pareto efficiency* of the method's solution set. The solution set is obtained by various possible configurations of that method against the two objectives. Pareto efficiency is based on the *non-domination* relation between solutions. A solution point x dominates y if and only if x is at least as good as y in both objectives and better in at least one [258]. A point x is *non-dominated* if and only if no other points dominate x. The set of all mutually non-dominated solutions is called the *Pareto front*. The experimental analysis aims to answer the following two questions:

q1. Does knowledge update lead to a high fidelity replica, such that the simulation model closely describes the real world and exhibits smaller utility deviation?

q2. Does knowledge equivalence checking achieve more Pareto efficient results than state equivalence checking under different uncertainties?

5.6.1 An Illustrative Example

The evaluation adopts a modified version of the real-time tracking by distributed smart mobile cameras[2] [74]. This example is one of the various studies where self-awareness has been applied and has demonstrated the advances in achieving self-organising behaviours [200]. In addition, the study of distributed smart cameras itself has also been actively studied regarding various decentralised approaches and different variations [75, 264, 193]. This example, which is based on [74], represents a typical setup for how networked self-aware agents in a spatial environment can self-learn by accumulating knowledge of interaction and self-organise to finish specific tasks. The example also fits this chapter's assumption of having decentralised agents that self-evolve their knowledge bases with interaction-awareness. This example case study is complex by exhibiting emergent properties: minor differences in the state will easily affect the behaviour of agents and diverge the self-organising behaviours of other networked agents thereafter.

In the example, a fixed number of objects and intelligent drones move in a bounded 2-D space. The drones are equipped with onboard cameras to collaboratively track and monitor objects for surveillance purposes. Each camera is assumed to have an omnidirectional (360-degree) view with a fixed sensing range. Each drone is controlled by an embedded autonomous self-aware agent which decides its actions. Each object moves towards a fixed direction at a constant speed, and will bounce back when it hits the boundary of the 2-D space. Objects can become important or unimportant spontaneously. The goal of the system of drones is to maximise *k-coverage*: trying to ensure every *important* object is monitored by at least k cameras. It is further assumed that as long as an object is located within the sensing range of any camera, the object is regarded as covered by that camera. An object is *k-covered* if it is covered by at least k cameras at the same time (see Fig. 5.3).

[2]All elements used for the experimental analysis presented in this chapter including the source code, parameters, data and results are available at http://digitwins.github.io

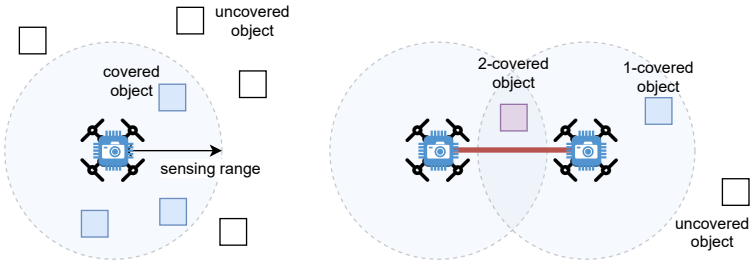


Figure 5.3: k-coverage from the bird's-eye view.

5.6.1.1 Utility of the Task Goal

To measure the satisfaction of task goal (*k-coverage*) over time, a *utility function* is used [74]. Suppose there are N_{all} objects in the environment. Let $N_k(t)$ be the number of objects being k-covered at time t. Then the utility at moment t is defined as the ratio of k-covered objects: $u(t) = \frac{N_k(t)}{N_{all}}$.

5.6.1.2 Self-Aware Agent

Since no single agent has a global view of the world (due to limited sensing capability), each agent is assumed to collaborate with others based on message communication. An agent can request other agents to help track the same object by advertising the object ID and position. Other agents can then decide whether to accept the request and move towards the advertised object. To avoid broadcasting to all other agents which is costly in communication, each agent is assumed **interaction-aware** to intelligently decide which ones to send messages to and which request messages to accept.

First of all, stimulus-awareness is the basis of self-awareness. The stimuli in the example are the local perception of the agent, which includes the received messages, the current position of itself, as well as the current positions and the importance of all objects within its camera range. An agent is able to use the stimuli to decide which direction to

move to track a certain object.

Beyond stimuli, the knowledge of interaction in this example is the status of co-covering. A drone is able to identify the coverage of other drones of objects as itself in a given time; this can be achieved through short-range wireless sensing technology as assumed in the original example [74]. Such knowledge of interaction is modelled as artificial pheromones, represented as a weighted graph, and maintained by each agent in its local memory. Agents are vertices in the graph. The edge weights reflect the recency of co-covering collaboration. If any two agents are both covering one or more same important objects, then the weight of the edge connecting the two agents will be "strengthened" by adding a constant value δ. All edge weights will also "evaporate" through time by multiplying a discount factor γ at the end of every time unit [74, 75].

With the knowledge of interaction, an agent tends to communicate with agents that it has collaborated with very often recently. When requesting help, the agent shall only notify the $(k + 1)$ agents with which it has the strongest weights. When deciding which request to accept, the agent will rank the received messages first by edge weight and then by the received time. The agent will only accept the most recent request sent by the agent that has the highest edge weight. A more detailed description of the behaviour of a single agent is shown in Algorithm 5 in appendix A which is based on [74]. Each agent in the system is designed as Algorithm 5.

5.6.2 Prototype Implementation

Two simulators are used to implement the example and its DT, as shown in Fig. 5.4. The two simulators run in parallel, denoted as *Simulator* ***P*** and *Simulator* ***D*** to represent the **P**hysical world and its **D**igital twin replication, respectively. Both simulators are implemented by the

agent-based simulation engine Repast Simphony [184], and can simulate the behaviours of objects and drones. The two simulators are identical `jar` executables, but multiple types of uncertainties are introduced between the two, which is explained later in section 5.6.3.1. The simulator initialises its starting scene by loading an XML file, which contains the information of all drones and objects as shown in Table 5.2. The simulation state σ_t as defined by Equation 3.3 is composed of the positions (x-y coordinates) of all objects and drones at time t.

The equivalence manager is implemented in Python and uses Py4J[3] to interact with the two simulators. It is able to sense data listed in Table 5.2 from both simulators at every time step. When an update of the DT is needed, the equivalence manager re-initialises *Simulator D* by passing all data in Table 5.2 sensed from *Simulator P* as an XML file.

Table 5.2: Types of data accessible in simulation.

Entity	**Attribute Data**
Drone	ID
	position
	received messages
	interaction graph
Object	ID
	position
	direction
	importance

5.6.3 Experimental Frame

5.6.3.1 Evaluation Scenarios

To emulate the situations in real-world settings, biases and uncertainties are considered. Following on the discussions of section 5.3.2, the following three threats to equivalence are

[3]https://www.py4j.org/

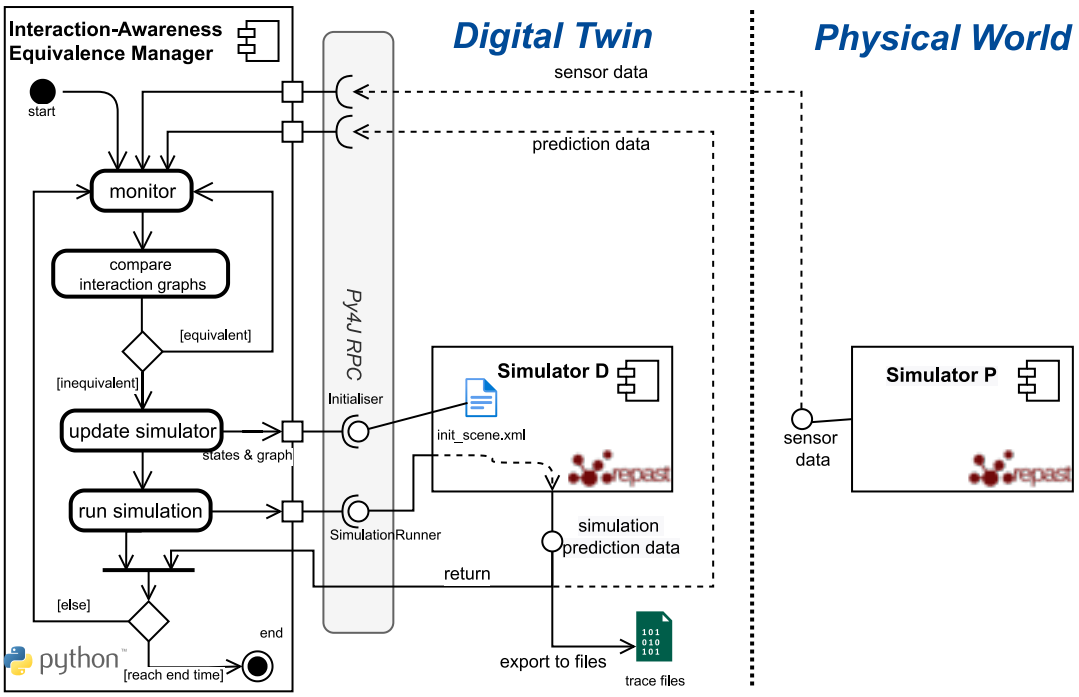


Figure 5.4: The implementation architecture for the prototype.

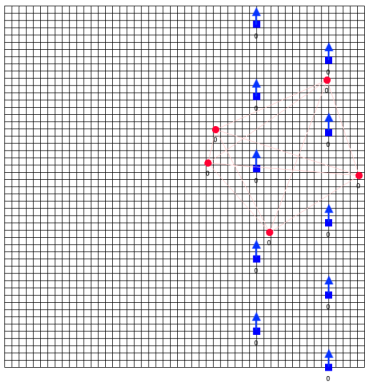
considered:

T1. Unpredictable environment changes: biased model parameters.

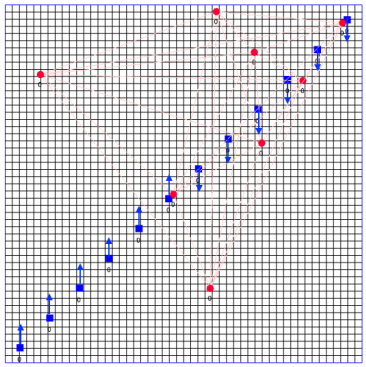
T2. Unpredictable environment changes: the environment is partially observable; data out of the sensor range are unknown.

T3. Nondeterministic agent behaviours.

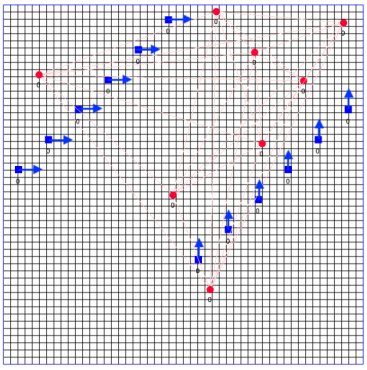
First, for T1, systematic deviation of an object could be encountered in the physical environment, but may not be considered in the model of the object. In the experiment, the simulation model assumes its objects to have experienced systematic deviation in movement, where each physical object moves with a systematic error angle of 3 degrees to its left, but the model is not aware of such a deviation. Such an error can be common. For instance, if the object moves with wheels, then the error could be caused by the manufacturing imprecision of the wheels.



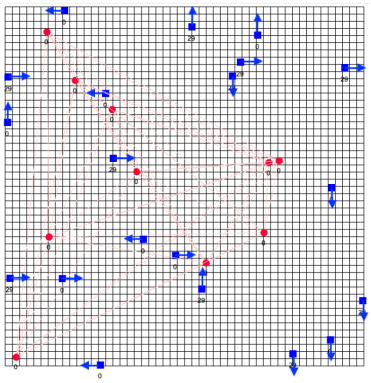
(a) Scene 1: 10 objects, 5 drones.



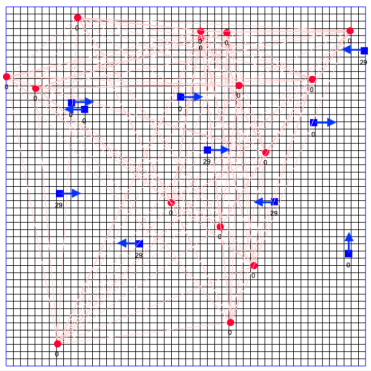
(b) Scene 2: 12 objects, 8 drones.



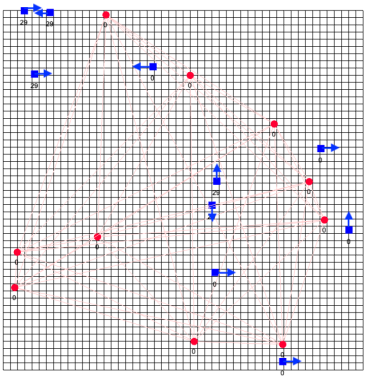
(c) Scene 3: 12 objects, 8 drones.



(d) Scene 4: 20 objects, 10 drones.



(e) Scene 5: 10 objects, 15 drones.



(f) Scene 6: 10 objects, 10 drones.

Figure 5.5: The initial snapshot of the evaluation scenes. A 2-D world of size 50 * 50 is considered. The blue rectangles are the objects. The red circles are the drones. Scenes 4, 5 and 6 are generated by randomly positioning the objects and drones and randomly assigning the initial moving direction of each object to be one of the four: east, north, west and south.

Second, for T2, if the DT is only able to acquire data from the drones in the physical world, then the positions of objects that are not covered by any drone are deemed to be unknown. Therefore, the behaviour of the entire system cannot be simulated accurately. These unknown positions can be estimated (e.g. through past trajectories of objects), but the estimation techniques are out of the scope of this chapter. In the experiments, it is assumed that the DT can estimate these positions with a random error ϵ on both x and y coordinates.

Third, for T3, the virtual agents and physical agents may exhibit different behaviours. When randomly deciding which object to track at line 11 of Algorithm 5, the physical agent and its virtual counterpart may make different choices. In the experiments, the virtual agents and physical agents are provided with different random seeds.

Two conditions are constructed by combining two threats. Each of the conditions, I and condition II are applied to six different starting scenes (shown in Fig. 5.5) respectively: (a) Condition I: T1 and T3 (b) Condition II: T1 and T2. Therefore, in total $2 \times 6 = 12$ different evaluation scenarios are designed. Each scenario is denoted by its condition and initial scene. For instance, scenario I-3 refers to the combination of Condition I and Scene 3.

5.6.3.2 Simulation Setup

The simulation uses discrete logical time. The model parameters for the two simulators are set as follows. The environment is designed to be a 50×50 2-D world with boundaries. The task goal is to maximise 2-coverage, with $k = 2$. Each object moves 1 unit distance every time step and becomes important or unimportant every 30 time steps. The sensing range of each camera is 10. The parameters of edge weights are $\gamma = 0.9$ and $\delta = 1$. The random error ϵ of position estimation described in section 5.6.3.1 is uniformly sampled from [-5, 5]. This

is a relatively large error since the uncertainty spans 20% of the side length of the 2D world, and it is at the same scale as the sensing range of the camera. Such an error assumes poor estimation of the unobserved environment by the DT. In the beginning, the two simulators are initialised with identical states and zero knowledge. Each experiment configuration is run for 1000 simulation time steps and repeated 5 times if not specified otherwise. All the experiments in this chapter were conducted on a MacBook Pro (13-inch, M1, 2020) laptop with macOS 12.5.1, Apple M1 processor, 8GB RAM, and 512GB SSD.

5.6.4 Simulation Validity of Knowledge Updates

Firstly, the performance of knowledge updates is evaluated. The aim is to investigate how different knowledge update strategies, when applied at different time steps of the simulation, can affect utility deviation over time. The evaluation is intended to investigate the question **q1** for the necessity of knowledge synchronisation. In particular, three update strategies used for synchronising the DT are evaluated, each differs in how the knowledge is updated:

- *Keep knowledge*: update all data items listed in Table 5.2, except for the interaction graph, and keep the graph in the simulated system unchanged.
- *Clear knowledge*: update all data items in Table 5.2, except for the interaction graph, and set all edge weights to zero.
- *Update knowledge*: update all data items listed in Table 5.2

Keep knowledge and *Clear knowledge* are two baselines. They represent two setups where the model update does not pay attention to the knowledge. Then the knowledge either remains unchanged or is reset. Only state variables and essential attributes of agents are synchronised to the DT. Instead, *update knowledge* synchronises not only the latest state

but knowledge to the DT.

Each experiment is designed to run for 100 time steps, which contains two phases: *knowledge accumulation* phase and *evaluation* phase. Each phase lasts for 50 time steps. The beginning of *knowledge accumulation* is called the *start time*, and the end is called the *update time*. At *start time*, the simulated system is initialised by replicating the snapshot of the real system, which contains all data listed in Table 5.2. At the *knowledge accumulation* phase, both systems run for 50 time steps to allow their knowledge to evolve. The two systems also gradually deviate from each other because of condition I. Next, at *update time*, the three update strategies are applied respectively to the simulated system by sensing the real system. Finally, at the *evaluation* phase, both systems run another 50 time steps, and the utility deviation is evaluated.

To emulate updating at different points in time, for each of the scenarios I-1 to I-6, 30 time stamps are randomly sampled from 1 to 900. Each time stamp serves as the *start time* for one experiment. Each experiment is repeated 10 times.

The average results for $30 \times 10 = 300$ runs of each scenario are shown in Fig. 5.6. The deviation of each bar is relatively large because even for one certain scenario, the system behaviours during different time periods are different. Then the performance of utility deviation may fluctuate among these 30 randomly sampled time periods. Nevertheless, by taking the average, the figure shows that for all the scenarios, the *update knowledge* strategy performs the best by achieving the lowest utility deviation. If knowledge is kept the same as before, the old knowledge can still take effect but the accuracy of utility is lower. If knowledge is reset at the update time, the simulation generally cannot guarantee the accuracy of utility as the other two strategies. Since different knowledge leads to different actions of the agent, the resultant task utility may also be affected. Therefore, when updating, in addition to maintaining the same state in the simulated system, the same knowledge should also be

maintained.

In addition, Fig. 5.7 shows the utility of scenario I-6 during the *evaluation phase*. The start time and update time are set as 270 and 320, respectively. The figure clearly shows that updating knowledge ensures the simulation most accurately "follow" the ground truth utility. Notice that the simulation starts to deviate only after time 333 in both three sub-figures. This is first because all objects switch their status of importance at time 330 (as assumed in Section 5.6.3.2). The important objects covered by drones then become unimportant. Therefore, the drones need to search and cover other objects that just became important, which leads to different movement patterns than patterns before time 330. Also, Condition I causes the positions of objects in the simulation to deviate from the real objects. With different drone movement patterns and different object positions, the utility deviates after time 333.

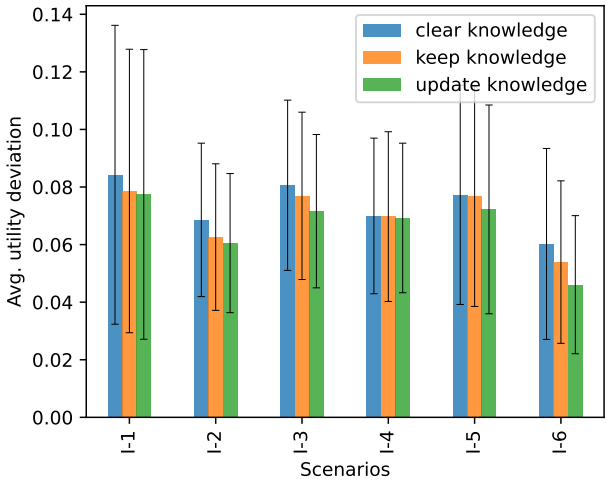


Figure 5.6: Knowledge update strategies in 6 scenarios.

5.6.5 Pareto Efficiency of Knowledge Equivalence Checking

Next, the proposed knowledge equivalence checking methods are evaluated: knowledge comparison and action comparison, which do not assume a fixed update frequency. When updat-

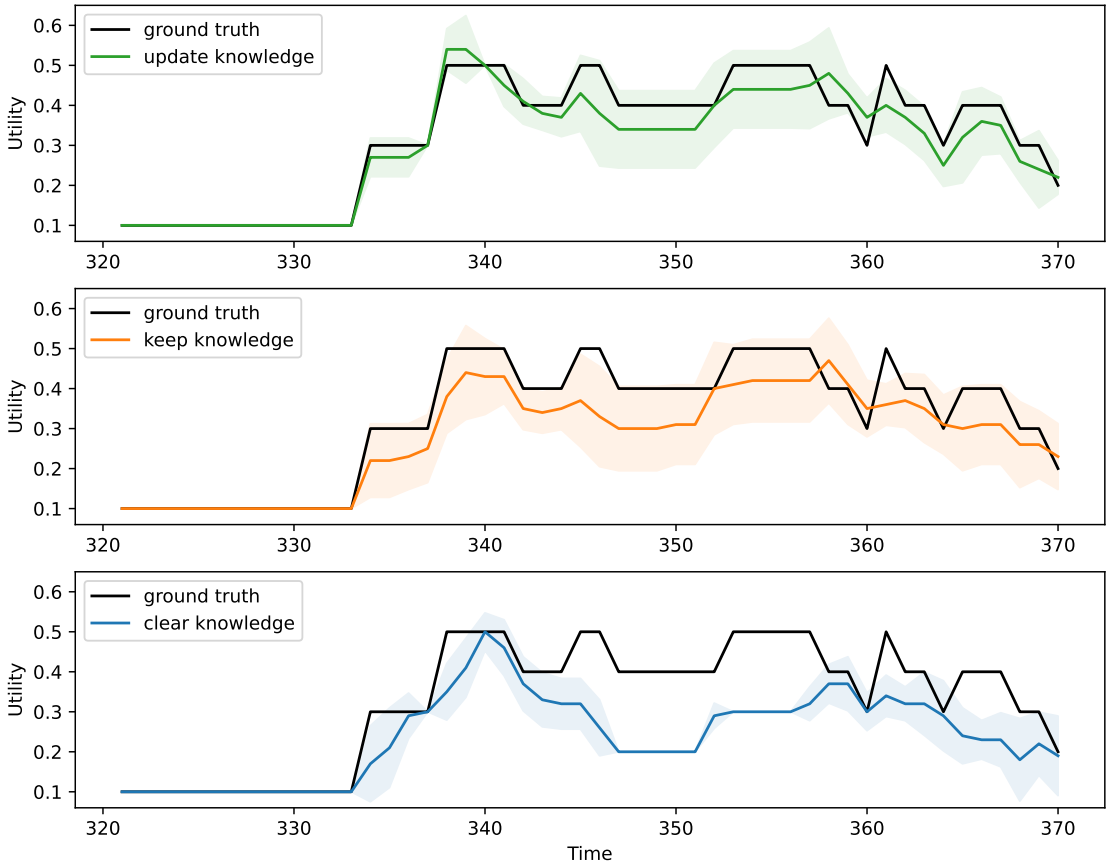


Figure 5.7: 2-coverage utility of three knowledge update strategies, after updating at time 320 in Scenario I-6.

ing Simulator D, all data in Table 5.2 (state and knowledge) are replicated from Simulator P. This evaluation is intended to study the question **q2**.

The method *state comparison* is regarded as the baseline, which is shown in Algorithm 3. A state is composed of the positions of all the cameras and objects. The deviation of two states is also measured by their Euclidean distance as:

$$deviation(\sigma_t, \sigma'_t) = |\sigma_t - \sigma'_t| = \sqrt{\sum_{i=1}^{l} (\sigma_{i,t} - \sigma'_{i,t})^2} \tag{5.9}$$

Algorithm 3: State Comparison.

Input: Comparison time interval q, threshold θ, comparison time window l
Data: History of physical world state $\mathbf{\Sigma}$, history of virtual world state $\mathbf{\Sigma}'$, time t

```
1  Σ ← [];
2  Σ' ← [];
3  t ← 0;
4  while True do
5      σ_t ← sense from simulation;
6      σ'_t ← sense from real world;
7      Σ[t] ← σ_t ;
8      Σ'[t] ← σ'_t;
       // Compare every q time ticks
9      if t mod q == 0 then
10         d ← 0 ;
11         for j = max(0, t-l) ... t do
12             d ← d + deviation(Σ'[j], Σ[j]) ;
13         end
14         if d > θ then
15             k_t ← sense knowledge of all physical agents;
16             Re-initialise simulation with σ_t, k_t ;
17         end
18     end
19     wait for the next wall clock time tick;
20     t ← t + 1;
21 end
```

There are three parameters for each comparison method. It is assumed that the comparison is made every time step $q = 1$ and the time window is $l = 1$. The performance

Algorithm 4: Fine-Grained Action Deviation.

Input: The action made by a physical agent, the action made by the virtual agent
Output: The deviation of two actions

if *both actions are* `random walk` **then return** 0;
if *both actions are* `follow` **then**
  if *both agents follow the same object* **then**
    return 0;
  else return 1;
if *both actions are* `respond-and-follow` **then**
  $d \leftarrow 0.4$;
  if *both agents respond to the same agent* **then**
    $d \leftarrow d - 0.2$;
  if *both agents follow the same object* **then**
    $d \leftarrow d - 0.2$;
  return d;
if *both actions are* `notify-and-follow` **then**
  $c \leftarrow$ the number of commonly notified agents by the two given agents;
  if *both agents follow the same object* **then**
    return $0.5 \cdot (1 - \frac{c}{k-1})$;
  else return $0.5 \cdot (1 - \frac{c}{k-1}) + 0.5$;
if *one action is* `follow` *and the other is* `notify-and-follow` **then**
  if *both agents follow the same object* **then**
    return 0.5;
  else return 1;
return 1;

of the comparison method is largely dependent on the threshold value θ, which defines the tolerance to deviation. This chapter evaluates various different values of θ for each method, and contrasts the three methods by their Pareto efficiency. Experiments for each threshold value are repeated 5 times.

Knowledge comparison and state comparison are both applied to each of the 12 scenarios. Action comparison is applied to scenarios II-1 to II-6. In action comparison, the action is viewed at a coarse grain and only 4 different actions are distinguished: `follow`, `notify-and-follow`, `respond-and-follow`, and `random walk`.

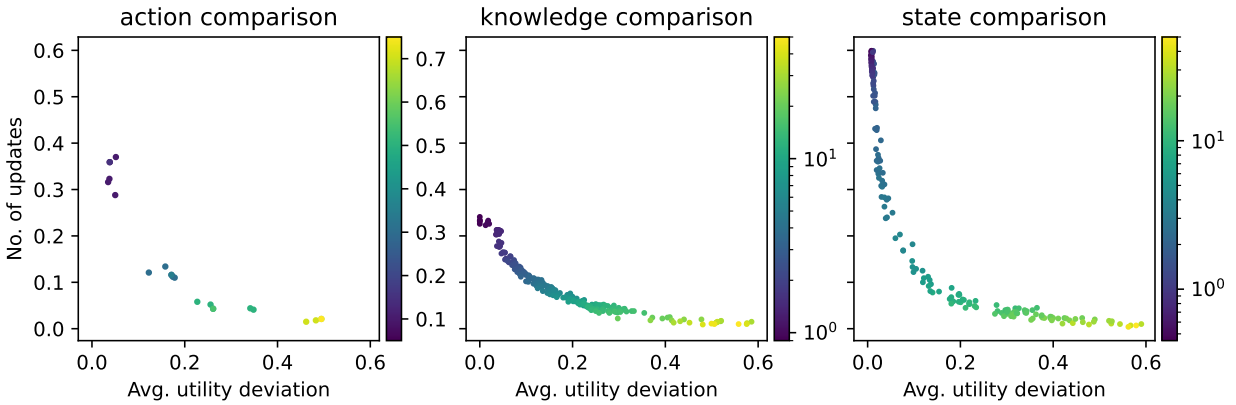


Figure 5.8: Solutions (after normalisation) obtained by different threshold values in scenario I-1. Values of the threshold are shown in different colors.

5.6.5.1 Performance Trade-off Due to Threshold Values

The different solutions obtained by various threshold values can be seen in Fig. 5.8, in which the three comparison methods are applied to scenario I-1. The trade-off between the two metrics can be observed for each of the methods. When the threshold value is small, the number of updates is large and the utility deviation is small. Conversely, a larger threshold will lead to a smaller number of updates but a larger utility deviation. This result validates the existence of a trade-off curve between the two metrics. Therefore, Pareto efficiency is then used to compare the solution sets of the three methods.

5.6.5.2 Analysis by Pareto Efficiency

The results for the experiments on all 12 scenarios are shown in Fig. 5.9. In all the 12 scenarios, knowledge comparison shows more Pareto efficient results when the average utility deviation is small, which are shown in the highlighted area of each sub-figure. A clear divergence trend can be observed in the solutions that as the average utility deviation decreases, solutions of knowledge comparison start to dominate state comparison. Knowledge comparison also exhibits a flatter curve in contrast to state comparison. In the extreme

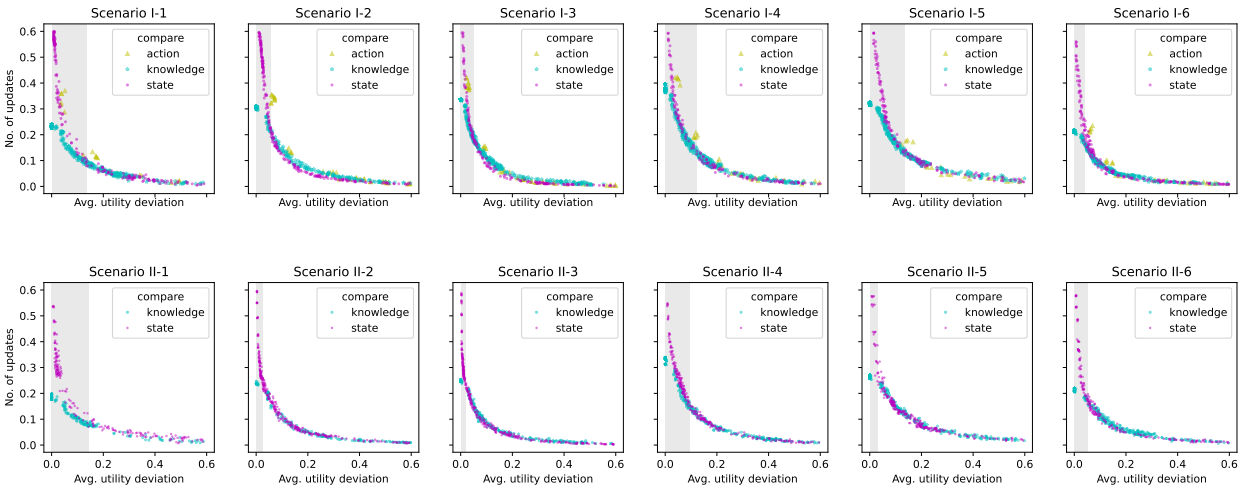


Figure 5.9: Simulation results for all 12 scenarios. The x-axis is normalised by the value of the average utility deviation obtained when no update is involved. The values used for normalisation are (row by row, from left to right for each row in the figure): 0.1436, 0.1452, 0.2224, 0.1343, 0.1165, 0.1416, 0.1519, 0.1519, 0.2167, 0.1449, 0.1250, 0.1475. The y-axis is normalised by 1000, which is the worst situation where the simulation should be updated each time unit. Within the highlighted area in grey, all solutions of state comparison are dominated by knowledge comparison.

case, knowledge comparison achieves 0 utility deviation with much fewer updates than state comparison. However, when the utility deviation is larger, state comparison can be sometimes better or similar to knowledge comparison. In addition, action comparison is generally slightly worse than both other methods.

Performance of knowledge comparison, as depicted in Fig. 5.10 reveals a common situation under threats T1: though the utility and knowledge are correctly simulated, the drones of the simulated world show complete deviation, when compared to their real positions. In this situation, two drones are tracking one object. In both the real and simulated worlds, the two drones are always covering the given object. The utility of 2-coverage for this two-drone system is always 1 in both worlds during all the 5 time steps. The knowledge in both systems is incremented the same way thus being identical. However, since the positions of the two systems are different, state comparison will regard the simulation as "deviated" and then will trigger an update, but knowledge comparison will not. Therefore, this situa-

tion shows that even with no utility deviation, state comparison can still invoke unnecessary updates, as observed in the highlighted areas of Fig. 5.9. Nevertheless, the advantage of knowledge comparison is only prominent when the tolerance by threshold is small. With larger thresholds, the simulation will evolve much further over time until the accumulated deviation is greater than the threshold and triggers the next updates. After evolving longer over time, the complex interaction with other drones and objects may lead to an entirely different system topology in the simulation, where the knowledge, state, and utility are all largely deviated. In this case, monitoring state (position) deviation might be more relevant to utility deviation, since the utility of coverage is calculated based on the relative position between objects and agents. This phenomenon is shown by the non-highlighted areas in Fig. 5.9. Also, if contrasting conditions I and II, the highlighted areas in II are generally smaller than I. This is because the large inaccuracy of the positions of uncovered objects under condition II causes the knowledge and system topology to deviate faster than condition I. For condition II, only when the threshold is smaller when compared to I, knowledge comparison can outperform the state comparison.

Therefore, when the requirement of the DT application focuses on minimising the utility deviation of the simulation, knowledge comparison can reduce much more overheads caused by unnecessary updates. With the same number of updates, knowledge comparison also ensures a more accurate simulation of the task utility when compared to state comparison.

This chapter then quantitatively compares the three approaches by the notion of *hypervolume*, which measures the quality of a Pareto front by the size of the space dominated by all nondominated solution points with respect to a reference point [258]. A larger hypervolume is better. This chapter uses $(1, 1)$ as the reference point for hypervolume calculation. The results are shown in Table 5.3. The values in the table confirm the observations above for the three approaches: in almost all the cases, knowledge comparison has a larger

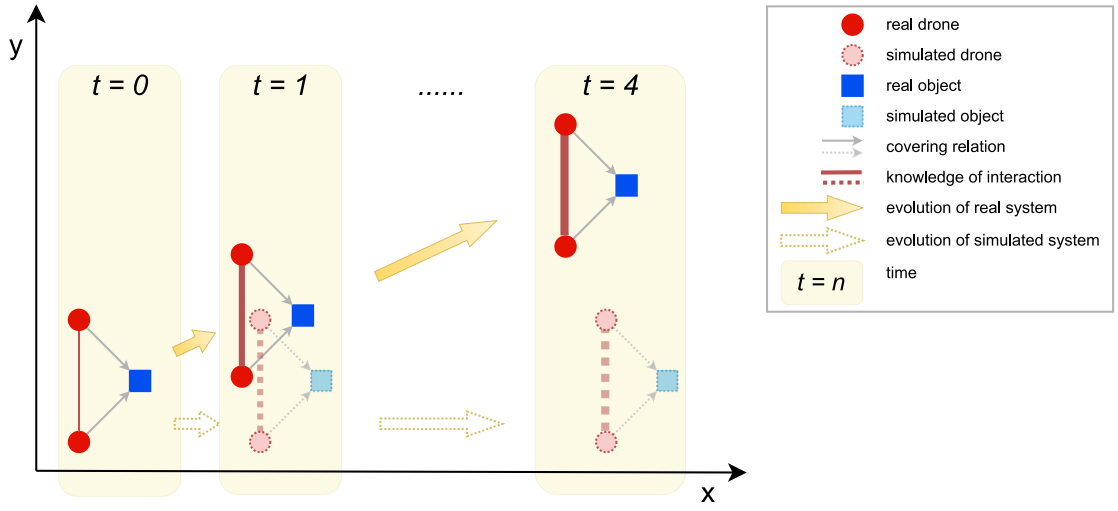


Figure 5.10: A situation where the state deviates but knowledge and utility are not. At t=0, the simulated world and real world are identical, hence they overlap at the same position.

Table 5.3: All solutions measured by the hypervolume of the two objectives: utility deviation and the number of updates.

Methods	Test Scenarios					
	I-1	I-2	I-3	I-4	I-5	I-6
State	0.9503	0.9456	0.9638	0.9401	0.9231	0.9649
Knowledge	**0.9629**	**0.9500**	**0.9643**	**0.9489**	**0.9385**	**0.9699**
Action	0.9090	0.8910	0.9336	0.9013	0.8417	0.9104
	II-1	II-2	II-3	II-4	II-5	II-6
State	0.9525	0.9607	**0.9721**	0.9458	0.9407	0.9562
Knowledge	**0.9635**	**0.9638**	0.9711	**0.9542**	**0.9469**	**0.9646**

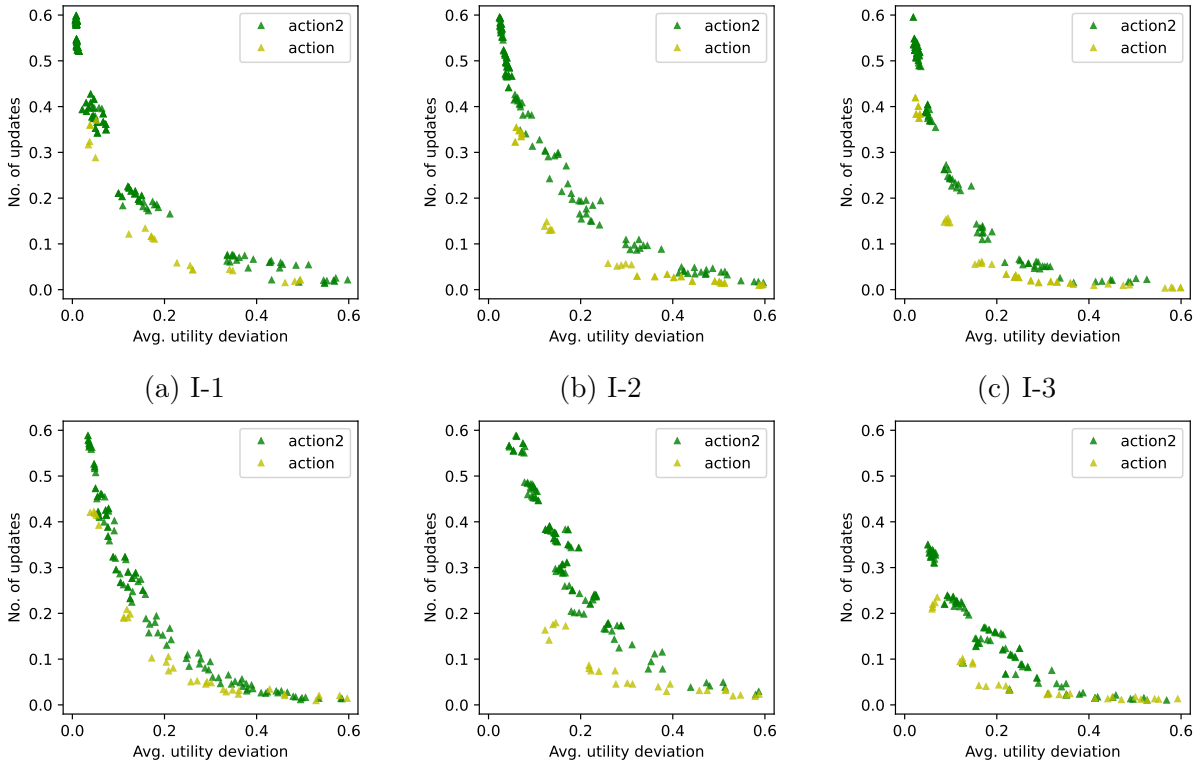
hypervolume than state comparison; action comparison has the smallest hypervolume.

Additionally, in action comparison, the action and its deviation can be quantified by viewing it at different levels of granularity. The previously evaluated view of the action is regarded as the coarse-grained view. A fine-grained view of action deviation is then further designed and evaluated. The calculation follows Algorithm 4, which defines the deviation of the actions of one physical agent and its virtual agent counterpart at a particular time step. The evaluation is based on scenarios I-1 to I-6 and follows the same procedure as Section

5.6.5. The results are shown in Figure 5.11, in which the newly introduced fine-grained view is denoted as `action2` and the coarse-grained view is denoted as `action`. According to the results, the coarse-grained view `action` is more Pareto efficient in all 6 scenarios. One possible reason is that the `action` is more related to the utility deviation. The utility is measured by the number of objects being covered in a global view, not who is covering which specific object. The four coarse actions used by `action` indicate whether an agent is covering any important objects, which directly contributes to the utility of k-coverage. For instance, if the virtual and physical agents are both doing the action `respond-and-follow`, then this means they have both not sensed any important object. Then `action` will regard them as equivalent, because responding to which particular drone and trying to follow which object does not make any difference in k-coverage at the current moment. However, `action2` would distinguish in detail about the other drone and object, which can be too strict in specifying deviation.

5.6.6 Memory Usage of Knowledge Equivalence Checking

The memory usage of each method is evaluated, and the results are shown in Fig. 5.12. The y-axis of the figure is the total extra amount of sensor data that needs to be loaded into the memory for comparison. Action comparison uses much less memory when compared to the other two, since an action can be encoded as one integer. Despite being a bit less Pareto efficient than the other two methods as in Figure 5.9, action comparison is the best choice when less memory consumption is preferred. Although knowledge comparison is more Pareto efficient in the previous evaluation, its memory consumption will increase exponentially as the graph size increases. For instance, scene 4 has 10 cameras and 45 graph edges, while scene 5 involves 15 cameras and 105 edges, causing memory consumption to double in scene 5 than scene 4.



(a) I-1 (b) I-2 (c) I-3

(d) I-4 (e) I-5 (f) I-6

Figure 5.11: Two different ways of quantifying actions applied to scenarios I-1 to I-6.

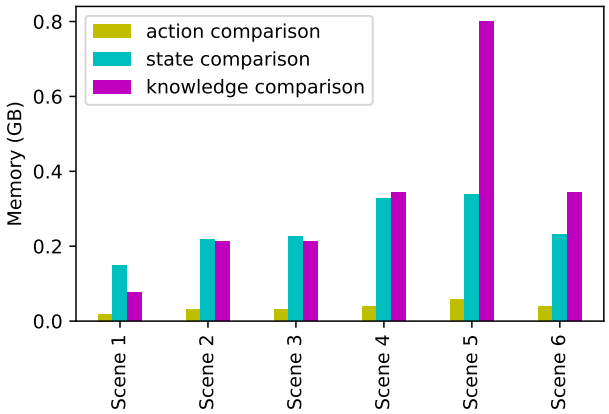


Figure 5.12: Extra memory required for the comparison.

5.6.7 Discussion

5.6.7.1 Evaluation Summary

The experiments have evaluated the effectiveness of knowledge update, and the performance of the two proposed knowledge equivalence checking approaches in terms of their update overhead, simulation accuracy and memory consumption. The evaluation is set given different types of threats to equivalence in section 5.3.2. The results have answered the two questions at the beginning of this section. First, for **q1**, maintaining knowledge equivalence is essential in keeping the DT model up-to-date, since it on average leads to lower simulation deviation of utility. Second, for **q2**, the proposed knowledge equivalence checking approaches have demonstrated their performance advances over the baseline (state comparison) in terms of Pareto efficiency. When a high simulation accuracy on task utility is preferred, knowledge comparison achieves the best trade-offs of reducing update overhead and increasing accuracy. When fewer updates are preferred, the three methods are generally comparable. Although knowledge comparison is sometimes slightly less Pareto efficient than the baseline approach specifically when utility deviation is large (e.g. in I-2 and I-3), such a situation violates the fundamental aim of the DT being an accurate representation. The quantitative results measured by the hypervolume metric also show that knowledge comparison is generally comparable to or better than the baseline. Although the proposed action comparison does not perform better than the other two methods in the Pareto efficiency of update overhead vs. accuracy, it can largely reduce memory consumption. Action comparison scales much better than knowledge comparison and state comparison in terms of the number of objects. Action comparison is most suitable in situations where a large number of objects are involved while the number of drones is relatively small.

The proposed knowledge equivalence-checking approaches can also be promising in other application domains. An indicative example is mobile edge computing. The mobile

devices offload some of their computation to the edge servers (e.g. base stations) nearby. Each edge server can leverage self-awareness to accumulate knowledge of the behaviour patterns of the surrounding mobile devices (e.g. latency, urgency of requests, computation capacity of the device) and past interaction with other edge servers. Based on this local knowledge, each edge server can thereby automatically decide whether to accept or migrate the offloaded computation in a load-balanced way. A DT of the entire edge-device system can provide a global view for more optimised offloading with what-if simulation analysis, but it may be infeasible to keep equivalent environment (mobile devices and their users) states every moment due to highly dynamic user behaviours. Nevertheless, knowledge is an edge server's world model and based on which the server takes actions for macro-level objectives (e.g. overall user satisfaction, quality of services). Knowledge has a closer link with these objectives than the state does. Therefore, using knowledge equivalence checking can tolerate state deviations that do not cause deviations in macro-level objectives, thus reducing the frequency of updates needed.

5.6.7.2 Threats to Validity

The experiment's validity can be compromised by several threats that arise from potential bias in the case study and the assumptions made for the experiment design. Identifying and addressing these threats is crucial for ensuring the reliability of the results, and these threats are presented below.

1) Real-World Data and Case Studies Controlled experiments have been conducted using simulated data. Simulated data have provided us with a cost-effective path for learning about the behaviour and performance of enacting knowledge equivalence in DTs under a range of scenarios that can exemplify complex real settings. Nevertheless, further systematic studies will consider data from real-world systems and domain-specific properties to evaluate

for knowledge equivalence.

The validity of the evaluation case study is made possible with two implicit assumptions: first, minor differences in the state are indistinguishable when abstracted as knowledge, and will not change the behaviour pattern of the drones; and second, knowledge deviation is more sensitive to changes in task utility, compared to state deviation. Therefore, the state imprecision in the DT can be tolerated to some extent, and will not cause deviation in task utilities between the simulated world and the real world. Differences in knowledge indicate that a prominent deviation possibly has happened in the state, and it is also why the knowledge has drifted. Only example cases that share the same assumptions above can observe a similar improvement in performance by knowledge equivalence checking. Even in the evaluation, these two assumptions do not hold all the time, which explains only under the preference for minimising deviation, knowledge comparison can be more Pareto efficient than state comparison.

2) Types of Threats to Equivalence In the real world, different types of uncertainties exist. In the experiments, only three types of threats to equivalence are considered. Further studies are needed to evaluate for the manifestation of multiple types and/or compounded threats that may exist at the same time, including the threats mentioned in section 5.3.2. In addition, the controlled experiments assume a situation where the configuration of the environment is known: the number of objects and how an object will become important is known. Further studies need to consider highly volatile dynamic and opportunistic scenarios.

3) Simplification in the Evaluation Model This chapter adopts a simplified model for calculating network and computation overhead. The overhead caused by network communication is not studied. For systems consisting of many drones and objects, data transmission to the DT may become a bottleneck and can increase drone energy consumption. The time

required for data loading and processing in the DT is not modelled. Instead, this chapter measures the overhead caused by data loading and processing by the "number of updates". To achieve real-time or faster-than-real-time simulation, the model must match the application regarding simulation time scale, model fidelity, and computation efficiency. This chapter assumes that the DT can run simulations in real-time, but further research is needed to investigate time limitations when running simulations that must meet time constraints for prompt decision-making.

5.7 Summary and Future Work

This chapter has investigated the problem of knowledge equivalence in the context of digital twins (DTs) of intelligent multi-agent systems. The objective of the work is to ensure that knowledge equivalence and self-awareness are kept up-to-date in the DT and synchronised with the physical world. Knowledge equivalence is particularly important for unlocking cognitive capabilities in the DT that can promote new opportunities for sophisticated analysis (e.g., what-if prognostic and diagnostic analysis) that can further support and enhance intelligence in the physical world. Additionally, the contribution of this chapter shows that knowledge can provide an alternative equivalence metric for such a complex setting, where using knowledge equivalence can alleviate overheads of continuous calibration of models, which leverage state, and/or input parameters comparison and updates.

This chapter has presented a conceptual architecture for the intelligent self-aware DT and an approach for checking knowledge equivalence at the interaction level. Analysis of the proposed approach demonstrates that knowledge equivalence can tolerate deviation thus reducing unnecessary updates and unwanted overheads. When compared to discrepancy analysis based on state, knowledge equivalence can achieve more Pareto efficient results on

the trade-off between overhead (number of updates) and effectiveness (utility deviation). This can significantly reduce update overheads, when lower utility deviation is preferred.

Chapter Six

Virtual-to-Physical: Digital Twin-Enabled Adaptation for Self-Aware Systems

The P2V equivalence maintenance mechanisms in the previous chapter ensure a valid model on which prescriptive simulation analysis can provide valid predictions. This chapter focuses on the other dimension of the mutual intelligence enrichment of the feedback loop: V2P. In particular, given an equivalent model, this chapter investigates how a DT can utilise its cognitive capabilities to enhance the intelligence of the physical system with more informed online simulation analysis. This reflects the meta-level analysis mentioned in Section 3.3.1. this chapter particularly focuses on the DT's ability to adapt the decision function π of the physical agents. This chapter takes the first step in discussing the inter-relationship between the intelligence of the DT and the intelligence of the physical system in the V2P direction. In this chapter, the physical system is assumed to be at the basic level of self-awareness: **stimulus-awareness**. The physical agents are instantiated with static rule sets to reactively alter their behaviours based on external events. This chapter focuses on **goal-awareness** capability in the self-aware DT reference model proposed in Figure 4.2, and apply the **goal-**

aware DT to improve the rule sets with more informed online simulation analysis.

This chapter specifically set the problems in the domain of adaptive compliance in Cyber-Physical Systems (CPS) with trade-off analysis to be made online. The holistic architecture is refined according to the domain with related analysis techniques and models. A case study in smart warehouse logistics is studied to demonstrate the refined architecture and the proposed approach.

6.1 Overview

Regulatory compliance is "the act of ensuring adherence of an organisation, process or (software) product to laws, guidelines, specifications and regulations" [5]. This chapter specifically focuses on the compliance for software products. Emerging digitalisation trends in application domains like smart manufacturing and smart logistics emphasise the necessity of compliance with various regulations and standards (e.g., General Data Protection Regulation [GDPR], ISO series, safety, security, etc.) [136, 268]. In these application domains, the systems under study may consist of numerous smart entities, such as autonomous mobile robot swarms, internet-of-things, etc. This presents new challenges for the design of the system to be in compliance with standards such as safety [113]. These systems are essentially complex CPS. They often operate in open and dynamic environments, facing unpredictable behaviours and sometimes involving interaction with humans, making it a challenge to realise the desired level of compliance (e.g., safety).

6.1.1 Challenges

Autonomy is an important dimension in CPS that enables the system to perform specific tasks in dynamic and complex environments without human intervention [174, 39, 140]. Compliance with regulations such as safety needs to be guaranteed for autonomous behaviours in the face of different environments at runtime. These autonomous behaviours are enabled by software, which poses requirements for designing software that enables safety compliance behaviours at runtime. Engineers need to design the software-enabled communication, computation and control scheme with the desired level of compliance in mind to ensure acceptance. For instance, in human-robot collaboration, safety is a major concern. Automatic motion control strategies need to ensure that humans are safe while working alongside robots [151].

Designing for compliance requires the mapping from compliance sources (specifications of regulations, standards, etc.) to software-regulated behaviours. However, for autonomous CPS in a complex dynamic environment, the following challenges may arise.

Compliance and quality goals In addition to complying with regulations, the system aims to achieve other system-level or business goals. One essential aspect is maximising the productivity of the CPS, such as improving package delivery efficiency in an automated warehouse. Nonetheless, conflicts often arise between meeting compliance requirements and pursuing other goals such as productivity, profit, and cost [16]. A trade-off analysis is necessary.

Limited design effort and knowledge Quality assurance engineers often take a "best effort" approach to map standards, guides and regulations to process, control, and infrastructure. However, this effort of mapping is often constrained by time, budget, and priorities,

and is often biased to the expertise of the analysts. Additionally, the mapping process is complicated by considering various dependencies and conflicts between standards, guides and regulations, making it difficult to ensure optimal compliance at design time. Therefore, this difficulty calls for dynamic and continuous evaluation approaches for compliance, considering interaction trends (whether direct or implied) of various CPS components and their interactions.

Dynamic runtime behaviour For complex CPS, testing the software-controlled behaviour at design time can be inadequate for compliance. The designed control policies may only be applicable in specific scenarios, since not all possible future threats can be foreseen during the design phase, and the full behaviour of the complex environment only emerges during operation [213, 238]. Complex CPS may consist of multiple sub-systems. The interaction between each sub-system increases the dynamism and uncertainty of the system. In addition, errors caused by both humans and machines can have a ripple effect on the system. These errors are best understood during run-time due to the lack of necessary contextual information during design time. Henceforth, runtime compliance monitoring becomes a necessity for the sustainable and dependable operation of the CPS, and as a mitigation strategy for preventing degradation in the system and avoiding Service Level Agreement (SLA) violations.

Legal interpretation Legal and regulatory documents tend to be ambiguous and describe general norms, which sometimes need interpretation [165, 88]. As mentioned by Sánchez et al, "it is hard for engineers to assess and provide evidence of whether a technical design is compliant with the law due to the gap existing between a legal document written in natural language and a technical solution in the form of a software system" [213]. In achieving a certain regulatory/standard compliance, different interpretations can lead to different design

alternatives of the software product, which impact differently on other quality goals, also calling for trade-off analysis [88, 216]. Due to the dynamism of the environment, control policies that were optimal in the past may later become obsolete, thus decreasing the satisfaction of quality goals in new unforeseen environmental contexts. Strategies that are optimal for a certain part of the system may also lead to compliance violations and goal degradation in other parts of the system.

Traditional offline design methods face difficulties in handling uncertainties that may only arise during runtime, as mentioned earlier. During the design phase, it is challenging to anticipate or account for every potential environmental context that may exist for system testing. Therefore, it becomes necessary to extend the analysis and compliance management for the autonomous software-regulated behaviour of the CPS from design time to runtime, with dynamic trade-off analysis.

Research in compliance has mostly been considered offline, with the investigation for runtime adaptive compliance emerging [86]. However, the modelling and analysis of adaptive compliance are mostly at higher abstraction levels, such as goals and with tools such as SysML [5, 216, 13]. Models at this level of abstraction alone may fail to support analysis for complex and open environments, whose behaviour can involve cascading effects due to the interaction between sub-systems. The behaviour of these environments can only be best anticipated by simulation models that capture the fine-grained detailed information about each sub-system. Also, there is a lack of extensive studies for adaptive compliance in a real-time fast-changing environment.

6.1.2 Contribution

The Digital Twin (DT) can be a promising solution to the above challenges in real-time compliance governance for autonomous CPS in highly dynamic environments. The DT contains high-precision model replicas of the real system/environment [233]. The core of the DT is Dynamic Data Driven Applications Systems (DDDAS), enabling real-time data sensed from the system to be continuously assimilated into the model to update the model for better fidelity and to inform more accurate runtime prescriptive what-if analysis and decision-making, which in turn affects the behaviour of the running system. Research is still in its infancy in utilising the DT's high-fidelity simulation model to support analysis based on fine-grained behaviour predictions and runtime what-if analysis.

This chapter specifically aims at applying a goal-aware DT for adaptive runtime compliance governance for the autonomous software-controlled behaviours of systems of systems with human-in-the-loop. To the best of the author's knowledge, no existing work uses faster-than-real-time *computation* supported by a continuously updated fine-grained simulation model (i.e. DT) for the runtime governance of compliance. There is still a lack of extensive investigation for the runtime feedback loop in the compliance context that enables autonomous model update, reasoning and self-adaptation. In particular, the following novel contributions have been made:

- A Digital Twin-based architecture for runtime compliance governance that incorporates abstract goal modelling and detailed agent-based modelling for runtime monitoring, what-if analysis and adaptive control.
- The utilisation of goal modelling for compliance, especially in how its refinement is linked to the design alternatives of agents.
- The runtime trade-off analysis for design alternatives on their satisfaction of compliance

and other quality goals with Pareto efficiency.

- The applicability of the approach is demonstrated in a case study of human-robot collaboration in smart warehouse logistics with dynamic trade-offs for safety compliance, production, and workforce constraints. The results show that the Digital Twin can provide engineers with insights for live evaluation and continuous refinements of the system's behavioural rules for better compliance.

The rest of the chapter is organised as follows. Section 6.2 provides the related work in adaptive compliance. Section 6.3 formulates the problem of adaptive compliance in the smart warehouse context. Then, the DT solution is proposed in Section 6.4. Section 6.5 describes the case study of the smart warehouse and the detailed compliance modelling adopted by this chapter. The experimental evaluation is presented in Section 6.6. Finally, this chapter concludes in Section 6.7.

6.2 Related Work

This section presents the related work in runtime compliance governance related to business processes and requirements engineering as well as the initiative of adaptive compliance.

6.2.1 Runtime Compliance Governance for Business Process

Compliance has been extensively discussed in business domains, especially in process-oriented business environments. Comprehensive systematic literature reviews can be found in [98] and [177]. In business processes, runtime monitoring and detection of possible compliance violations are essential, since it would be infeasible at design time to anticipate the satisfaction

of runtime aspects such as timing and resource assignment constraints [21]. Through interpretation, compliance requirements can be transformed into objectives and subsequent specifications as compliance rules or constraints. Then the monitoring can be made possible against the rules and constraints [159].

Many existing efforts have been made in designing architectural frameworks for runtime compliance monitoring and governance. For instance, [33] presented an integrated framework using Service-Oriented Architectures (SOAs) to support the whole compliance management lifecycle, including modelling, monitoring, displaying and reporting on violations. Various Domain-Specific Languages (DSL) and Business Process Execution Language (BPEL) are used in the modelling. In [21], the authors propose a runtime self-monitoring approach to business process compliance in cloud environments. They embed the monitoring logic within the process model without having an external monitoring component.

Runtime compliance monitoring is also enabled by specific languages for modelling compliance rules such as Compliance Rule Graphs (CRG) and extended CRG (eCRG) [160, 131]. In [131], visual compliance monitoring is made possible with eCRG to support violation detection and cause traceability.

Beyond the specific monitoring approaches, [159] proposed a systematic conceptual framework for comparing approaches for compliance monitoring. Ten compliance monitoring functionalities (CMF) are elicited that can characterise the capability of any compliance monitoring approach.

Compliance in business processes is related to regulating human behaviours in an organisation to follow the rules. In contrast, this chapter focuses on designing software-controlled autonomous behaviours that follow regulatory requirements.

6.2.2 Compliance and Requirements Engineering

Compliance for software products is commonly discussed and analysed in requirements engineering [198, 38]. Requirements engineers need to extract relevant requirements (e.g. rights and obligations) from the legal text and monitor the compliance of the software in its whole lifecycle [38, 188]. Such monitoring is also essential for complex software systems of systems [238].

The modelling of regulations from a requirements engineering perspective includes using logic, goal models, and semi-structured representations [188]. Goal-oriented requirements engineering (GORE) is a prominent approach for the analysis of stakeholder goals to facilitate the elicitation, analysis and elaboration of system and software requirements [137, 11]. Numerous notations and languages have been developed to support GORE, such as *i** [266], KAOS [59], NFR framework [178], etc. Goal models can represent not only the intent but also the structure of law [5]. With goal modelling, for instance, the work [112] utilises *i** and *Nòmos* to express the requirements and regulations, where stakeholders' inputs can be used to refine the compliance model. The work in [180] proposes an extended version of a goal- and actor-oriented modelling language called *STS-ml* tailored for GDPR-specific principles and concerns. The work in [186] proposed the concept of compliance debt as a kind of technical debt. Neglecting or not imposing compliance is modelled as a kind of compliance debt, and the debt is used as a decision factor. Combined with economics-driven approaches, they manage the risk of tolerating these obstacles that impede the satisfaction of compliance requirements. In addition, extracting requirements from legal documents can be automated by deep learning large language models such as BERT [211, 1].

This chapter adopts goal modelling, a modelling approach from requirements engineering to support requirements elicitation and runtime analysis of design alternatives in the DT.

6.2.3 Adaptive Compliance

Modern CPS involves highly configurable software and operates in dynamic environments, which poses new requirements for self-adaptive compliance management at runtime. Adaptive compliance in software-based systems has been motivated by García-Galán et al., who apply the Monitor-Analyse-Plan-Execute (MAPE) loop for runtime self-adaptation towards continuous satisfaction of compliance requirements in case of any variabilities [86]. Violation of compliance can come from the variability of compliance sources, the systems, and the operational environment [86]. There are some preliminary studies in adaptive compliance. For instance, the work in [216] uses Goal-oriented Requirements Language (GRL) to model the variability of the environment in relation to sources. Different design alternatives for compliance control can exist, each with its trade-offs [216]. However, there is still a lack of using detailed computational modelling for behaviour simulation prediction to tackle the challenge of runtime compliance management for complex systems of systems.

One perspective in supporting self-adaptive behaviour is *Models@Run.Time*. Runtime models are self-representation of behaviour, goal, and structure of the system: changes made on the model at runtime can be reflected in the real system, and vice versa [27]. Formal methods such as real-time model checking [183] and runtime verification [213] is a common approach for monitoring and verifying possible specification requirement violations based on the runtime model. However, formal methods suffer from scalability issues such as state explosion problems. Models in formal methods usually apply to closed systems whose total number of states is relatively size-limited. Applying these methods to open systems such as multi-agent cyber-physical environments can be challenging.

Table 6.1 shows the related work for adaptive compliance at runtime. Currently, research in runtime compliance governance is highly dependent on using abstract models such as goal models for compliance monitoring. However, the capability of abstract mod-

els is limited when analysing open and complex environments and systems with enormous amounts of different states and interactions. An example is smart warehouse logistics, where autonomous mobile robots collaborate with human workers for parcel management. With complex interactions between entities, the consequences of decisions and their influence on different goals (e.g. productivity) and the level of compliance (e.g. safety of each human) for the entire warehouse from a holistic view are not easily anticipated with abstract models. Therefore, to support runtime compliance analysis of this type of complex system, high-fidelity detailed modelling using agent-based simulation is needed with what-if prescriptive analysis, which is not adequately addressed in the literature. This chapters leverage the concept of DT and combine the usage of fine-grained simulation models and abstract goal models to support runtime analysis of compliance.

Table 6.1: Related work for adaptive compliance at runtime.

Reference	Scenario	Models	Prescriptive analysis	Complex system
[30]	Industrial internet of things	Metric model	×	×
[31]	System of systems	Metric model	×	✓
[216]	Platform-as-a-Service	Goal model	×	×
[86]	Platform-as-a-Service	N/A	N/A	×
[33]	Telecom services	Process model	×	×
[160]	Business processes in bank accounting	High-level abstract model with Compliance Rule Graphs	×	×
This thesis	Smart warehouse with autonomous mobile robots	Goal model and agent-based simulation model	✓	✓

6.3 Problem Formulation

To address the aforementioned challenges of section 6.1.1, this chapter adopts a scenario of smart warehouse logistics and formulates the problems that arise. The question this chapter

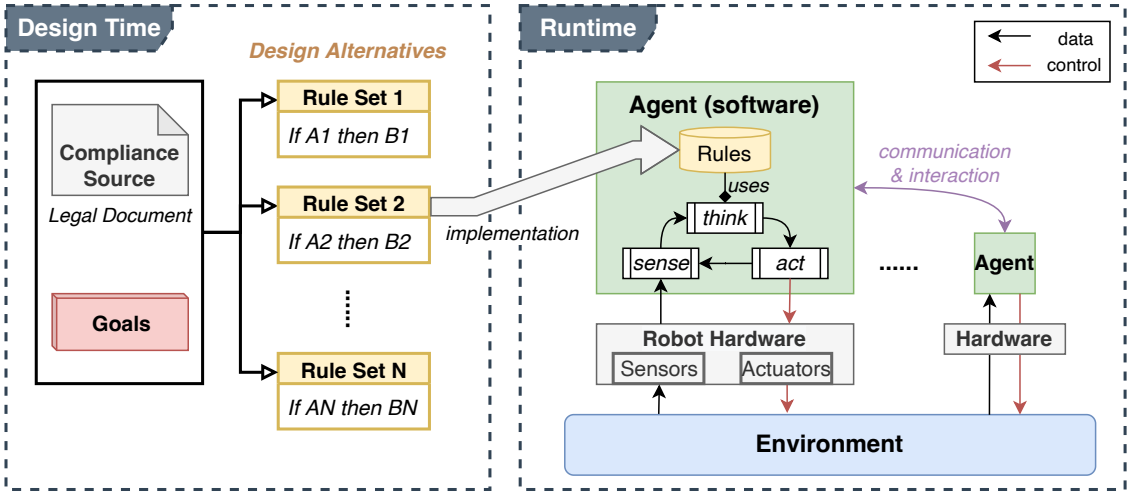


Figure 6.1: Design process: from compliance source to behavioural rules of agents.

is trying to investigate is: how a DT can overcome these challenges for the smart warehouse scenario? And how to model the problem and system using the DT approach?

The smart warehouse in this chapter is considered a multi-agent system consisting of Autonomous Mobile Robots (AMRs), collaborative robots (cobots), and human workers. Such a system is an autonomous complex system, with components (e.g., robots, humans, IoT sensors and intelligent processors, etc.) collaboratively working with each other or/and with humans to conduct logistics-related tasks. Each robot in the system is controlled by an onboard intelligent software agent that follows a *sense-think-act* cycle, as shown in the right part of Fig. 6.1. The agent uses sensors mounted on the robot to continuously perceive its surrounding environment, including information about other robots and humans, and tasks in the pipeline. In this chapter, each agent is assumed to follow specific behavioural rules designed to adhere to compliance requirements. One common type of compliance is safety, which requires agent-controlled AMRs to slow down or stop when human workers are nearby.

The challenge is to devise an adaptive tuning mechanism of the agents' behavioural rules under a highly dynamic environment, such that the smart warehouse system can 1) ensure regulatory compliance, such as safety while 2) maximising the satisfaction of quality

goals, such as productivity. The two objectives are usually conflicting, since a robot that is too safe may stop frequently even when a human worker is detected in its surroundings but far away from it, thus delaying the delivery of the robot's loaded package to the destination. In addition, as shown in Fig. 6.1, there can be multiple design alternatives for the rules of agents in the system, some of which can result from subjective interpretation of compliance sources. As discussed earlier, it is difficult for these design alternatives to be evaluated offline with full credibility, since compliance engineers are often challenged by limited knowledge, and possibly inflated, deflated or wrong assumptions at design time, especially for systems with novel situational and contextual use and new application domains. Additionally, some rules might not necessarily render compliance for all run-time scenarios, and contextualisation of these rules might be necessary to reach the desirable level of compliance. Consequently, it is imperative that behaviour rules for compliance need to be dynamically adapted at runtime to better cater for different scenarios and optimise for the trade-off between compliance and quality goals.

This chapter proposes to use DTs to tackle the challenge of the runtime trade-off between compliance governance and quality goals for complex systems of systems that operate in dynamic environments. In particular, this chapter aims to investigate the following questions: 1) how to utilise the DT approach to model the problem of runtime adaptive trade-off analysis? 2) How can the DT better explore the design space, and better understand the rules in different contexts?

6.4 Digital Twins for Adaptive Compliance

The Digital Twin (DT) is a paradigm that utilises high-fidelity computational modelling to create virtual replicas of real-world entities in the virtual world. The DT can leverage the

principle of Dynamic Data-Driven Applications System (DDDAS) [60] to utilise two-way communication between the real world and the virtual world to update its simulation model continuously and to provide simulation-informed decisions and control back to the real world (see Chapter 4). This section presents a DT solution for runtime compliance governance considering the trade-off with productivity goals, including the reference architecture, modelling and control.

6.4.1 A Reference Architectural Framework

This subsection describes a novel DT-oriented reference architecture for adaptive compliance, which builds on the holistic architecture (Figure 3.4 proposed in Section 3.3 to serve the compliance case.

The architecture is shown in Fig. 6.2, which mainly contains three parts: *physical space*, *simulation modelling and equivalence*, and *decision support*. The latter two parts constitute a novel extension to realise an intelligent compliance-aware DT.

The *physical space* contains robots (and their onboard controlling agents), human workers, and all other industrial assets. As mentioned before, the industrial system is viewed as a multi-agent system . Each agent has its own rules and can receive information and meta-level instructions (i.e. to adapt the behavioural rules) from the DT. The state and events of the environment and agents are continuously monitored by the DT in real time via various IoT sensors.

The *simulation modelling and equivalence* part is depicted above the physical space. Its core is an equivalent agent-based simulation model of the physical space that can forecast the behaviour of the physical space through what-if analysis for various scenarios. The simulation model replicates the environment (including the properties of humans, robot

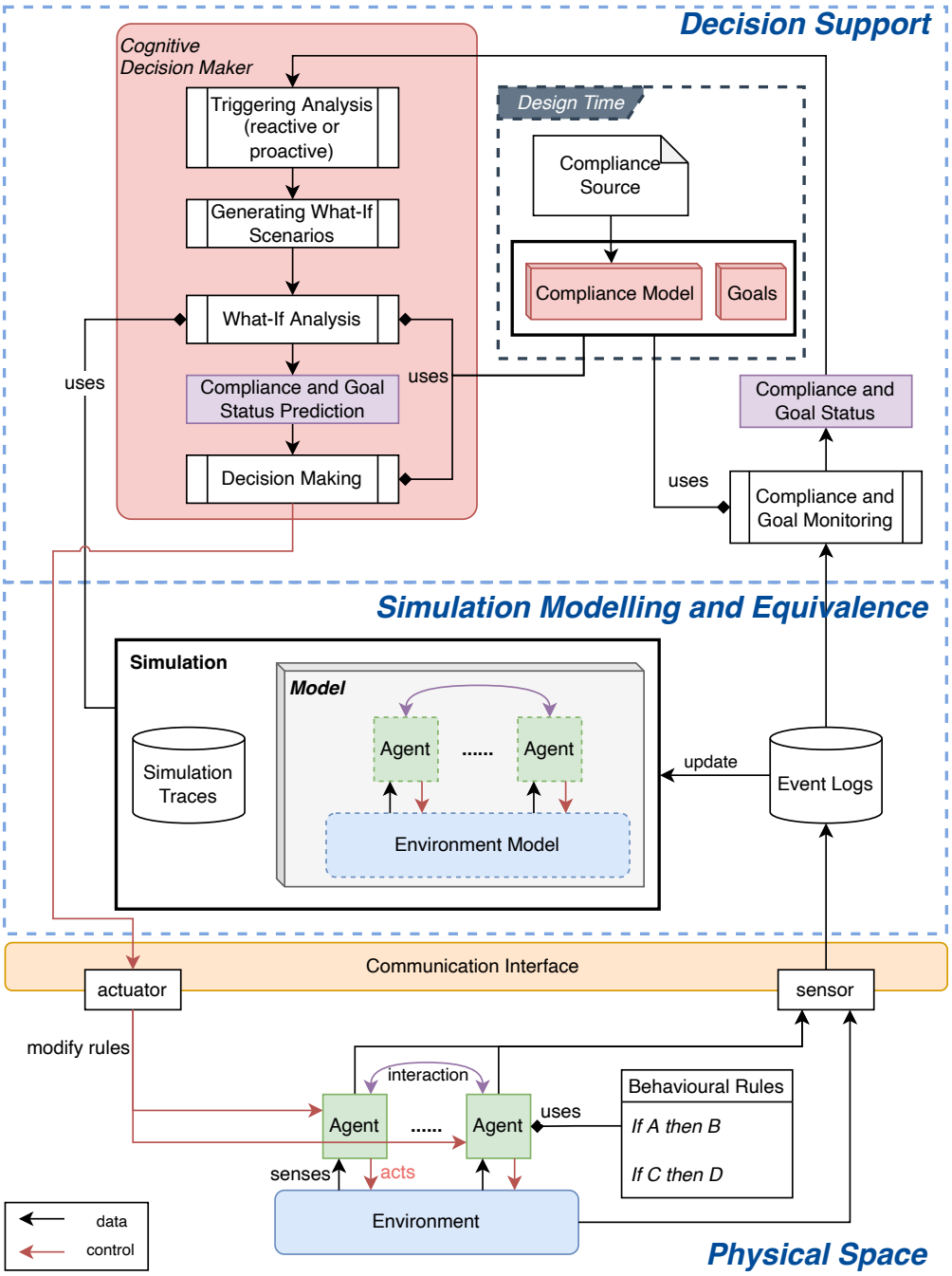


Figure 6.2: The reference architecture for a compliance-aware Digital Twin.

hardware, and other industrial assets) and the agents. The modelling of the environment can use a state-based approach where a set of state variables (e.g. locations of AMRs and humans) characterise the status of the environment at any particular moment. The evolution of the entire system model is controlled by other internal parameters (e.g. parameters in

the behaviour rules of AMRs). For the modelling of agents, since agents are essentially software programs, the "twinning" of the agents in the real-world AMRs can be achieved by replicating the same software program into the simulation model. The physical system's state and events are monitored and continuously assimilated into the model to perfect its fidelity for prediction. The agent-based model in the DT is updated in real-time in a data-driven manner through state replication and model calibration of the internal parameters.

The *decision support* part provides runtime analysis and optimisation for compliance and quality goals. This part monitors the status of compliance and goal satisfaction and informs further analysis. A cognitive decision maker is introduced; this novel addition utilises the up-to-date simulation model to run multiple what-if analysis at runtime by simulating the outcome of different design alternatives (as shown in the left part of Fig. 6.1) out of the same compliance source and productivity goals given the current state of the physical space. This analysis is essential since one certain design choice of the rule set may not be optimal in all possible situations during runtime. Such runtime what-if simulation analysis enables evaluating the applicability of different modifications of the agents' rules in real-time for trade-off concerns. The outcome evaluation also requires models that are able to represent the semantics of compliance and goals, and that can quantitatively measure the satisfaction of compliance and goals. In this chapter, Goal-oriented Requirement Language (GRL) is adopted for the modelling and refining of both compliance goals and specific requirements for the considered cases.

Finally, the candidate rule modification (one of the design alternatives elicited at design time) that leads to better trade-offs will be enacted in the agents in the physical space. Such an adaptation is done via *software* actuators of the DT only to modify the agent program.

6.4.2 Compliance Modelling and Assessment

Various approaches have been proposed for modelling compliance, among which goal modelling has been broadly studied. The advantage of goal modelling for compliance is that it addresses both the intent and structure of the law or regulations [5]. Many of the studies relate compliance with requirement engineering such that legal requirements and other system requirements can be aligned and reasoned altogether.

Goal-oriented Requirement Language (GRL) is a standardised language for modelling the objectives of stakeholders and systems, as well as their relationships. It can be used to elicit compliance requirements as goals and their solutions and obstacles, and represent them graphically as a tree-like structure. GRL supports quantitative and qualitative trade-off analysis for the satisfaction levels of the intentional elements (e.g., goals and tasks) and actors [11]. GRL has also been further extended with systematic guidelines for extracting and modelling legal statements [87].

This chapter uses GRL to model the compliance requirements and other system goals at design time. The satisfaction of requirements will be quantified by defining a set of metrics. Through goal refinement, GRL can provide a range of design alternatives for the rules of agents. Then, the goal model and metrics will be integrated into the DT-oriented architecture and utilised at runtime. Design alternatives not implemented in the agents will be evaluated at runtime using faster-than-real-time simulation to assess their satisfaction levels based on the metrics quantitatively.

Runtime assessment of these alternatives involves comparing them against multiple objectives (compliance and production goals). Pareto efficiency (or Pareto optimality) can be used for the comparison. Pareto efficiency is based on the *non-dominance* relation between solution alternatives, as shown in Fig. 6.3. A solution x dominates y if and only if x is at least

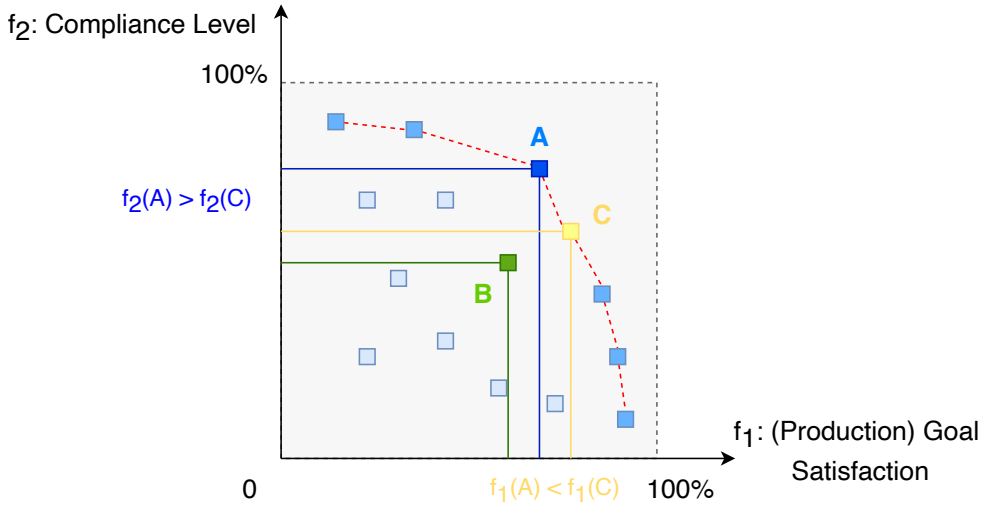


Figure 6.3: Pareto efficiency of two objectives: compliance level and goal satisfaction. A and C both dominate B, but A and C are mutually non-dominated solutions.

as good as y in all objectives and better in at least one [258]. A solution x is *non-dominated* if and only if no other solutions dominate x. The set of all mutually non-dominated solutions is called the *Pareto front* [258].

6.5 Case Study: Human-Robot Collaboration

This section presents a case study of human-robot collaboration in smart warehouse logistics to demonstrate the DT approach. The warehouse layout is shown in Fig. 6.4. Multiple AMRs and human workers work together in the warehouse to collaboratively pick and deliver requested items from incoming orders. Each order requests one item that is stored in one slot on a rack in the warehouse. Orders first arrive in a central dispatching system, and wait in queue until there is any idle AMR. When at least one AMR is available, the dispatching system will assign one order to one AMR. The AMRs are assumed not to be specialised in picking items off the racks, but only designed for ground transportation. For this reason, human workers will help in picking the items off the racks and loading the items onto the

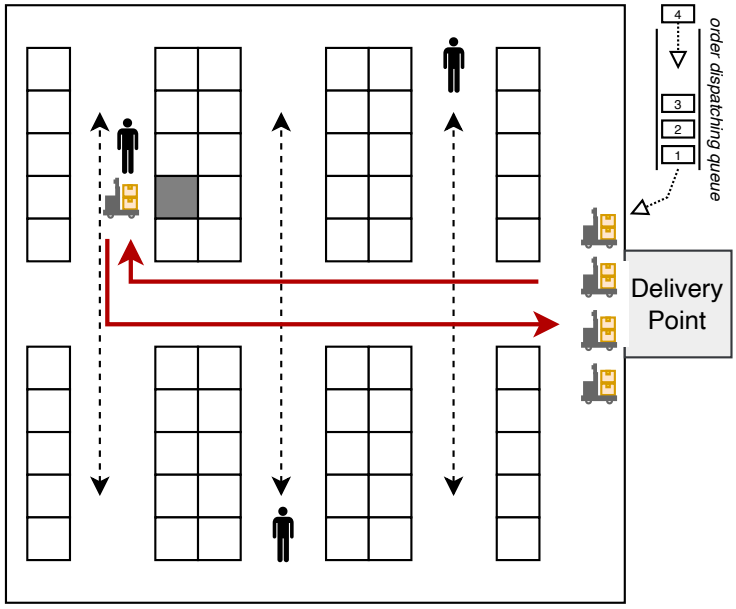


Figure 6.4: Overall layout of the case study. Human workers move according to black dashed arrows. The movement of AMRs (red arrows) may intersect with the workers. The safety compliance behavioural rule onboard an AMR should instruct the AMR to slow down to avoid collision with human workers.

AMRs.

The behaviour workflow of the AMRs and human workers is as follows. The AMRs are initially in an idle state and waiting for orders. After receiving the order information, each AMR needs to first plan its path to the location of the item. Then, the AMR notifies a human worker near that location by sending messages to the worker's smartphone. The AMR and the worker move to the location and wait for each other to arrive. The worker picks the item off the rack and loads it onto the AMR. Finally, the AMR carries the item and returns to the delivery point to finish delivery and then becomes idle.

For such a semi-automated warehouse, the number of human workers should be minimised to reduce the labour cost. This means that a limited number of human workers need to move to different locations in the warehouse to assist the AMRs, thus posing safety compliance requirements on the design of the AMR's autonomous movement control.

6.5.1 Compliance Modelling with GRL

This chapter focuses on safety compliance as a concern for this case study. This chapter also considers how this concern poses a trade-off against the productivity goal considering spatial requirements for safety and human-robot collaboration (individual and collective).

The safety compliance concern can be illustrated by Fig. 6.4. Workers need to move to different places within the warehouse, which may easily intersect with the paths of AMRs, causing collision and possible injury. Compliance with safety requirements for driverless industrial trucks has been specified in ISO 3691-4:2020, in which the *collision with person* is one of the main risks [113]. Specifically, in ISO 3691-4:2020 clause 4.8.2.1, detection of persons and collision avoidance are required: *"c) Personnel detection means shall be so designed that trucks shall stop before contact between the rigid parts of the truck or load and a stationary person (not a person stepping into the truck path or moving toward it) [...]"* In addition, such detection can be adaptive according to clause 4.8.2.6 (Selection of the active detection zone fields): *"Trucks can have an automatic selection of the safe detection fields based on truck speed and direction, size of the load or other criteria [...]"* It can be seen that the above requirements specified in the ISO document are ambiguous and general, which can be interpreted into different design alternatives and choices for the control in AMRs.

The GRL model of both compliance and production goals is shown in Fig. 6.5. The production goal is denoted as "increase productivity" and further refined as "reduce waiting time of orders". Safety compliance is regarded as a softgoal or non-functional requirement for each AMR, denoted as "safety of persons", which is further decomposed into two sub-goals: avoid collision and reduce speed. These two sub-goals specify two safety strategies to ensure the safety of the persons. They are connected to the top-level goal with an AND operator, meaning both two safety mechanisms need to be implemented. Each sub-goal is provided with a solution. The goal of collision avoidance can be sufficiently achieved by implementing

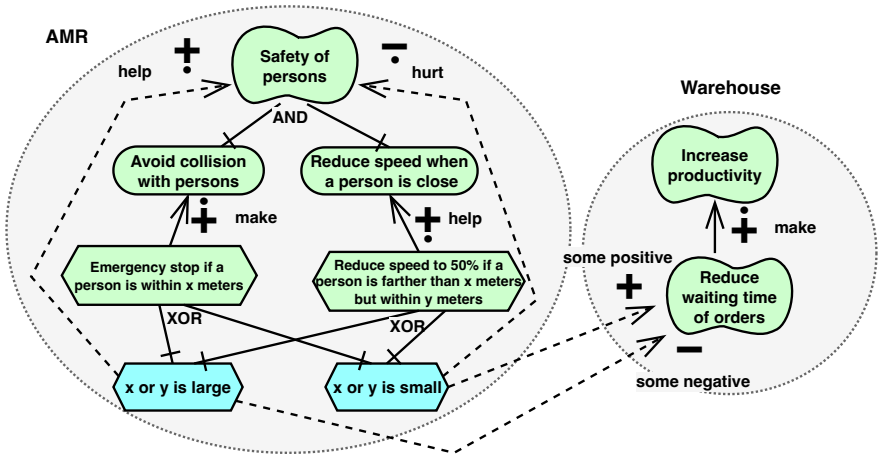


Figure 6.5: The GRL model for safety compliance and productivity.

a behavioural rule for emergency stops in the AMR, which is required by ISO 3691-4:2020 clause 4.8.2.1. For the sub-goal of speed reduction, it adds an extra level of safety without stopping the AMR's delivery task. In achieving the sub-goal of speed reduction, one solution is to design a rule for speed control with distance criteria, which is shown as a task element in Fig. 6.5: "reduce speed to 50% if a person is farther than x meters but within y meters". This task contributes to the satisfaction of its higher-level goal of speed reduction.

The distance criteria x and y in the two tasks (the two green hexagons in Fig. 6.5) can inform various design alternatives. Two examples of design alternatives are shown as sub-tasks in Fig. 6.5 denoted by cyan hexagons, indicating whether to use small or large values for the criteria x and y. The label XOR means only one of the two design alternatives will be chosen. These alternatives affect differently on the satisfaction of the compliance goal and the production goal, leading to a trade-off analysis. For instance, with a larger criterion, an AMR is safety compliant but may reduce its speed even when a person is too far away. In a busy scenario where a large number of human workers are involved, the AMR may reduce its speed too frequently such that its delivery can be delayed.

6.5.2 Metrics for Goal Satisfaction

This chapter then presents the design metrics to measure the levels of satisfaction of the two goals in the GRL model.

For safety compliance, first, a safety value is calculated for each person. Then the overall level of safety compliance can be measured by the mean safety value over all persons. The safety value of a person i is proportional to the distance between the person and the nearest AMR moving towards him/her. This distance is denoted as $d_i(t)$. Then the safety value of the person i is:

$$s_i(t) = \frac{d_i(t)}{\max\left(d_i(t), d_{th}\right)} \tag{6.1}$$

where d_{th} is a threshold: if $d_i(t) > d_{th}$, then the person is regarded as 100% safe meaning the AMRs are all too far away.

Suppose the warehouse involves in total N human workers. The overall safety level can be given in two ways using the mean or minimum value:

$$\text{Safety}_{\text{mean}}(t) = \frac{\sum_{i=1}^{N} s_i(t)}{N} \tag{6.2}$$

$$\text{Safety}_{\text{min}}(t) = \min_i \left\{s_i(t)\right\} \tag{6.3}$$

For productivity, the average service time of past n completed orders is used to measure the performance. Suppose at time t, the most recently completed order is the k-th one counted from the initial starting of the smart warehouse system, then the productivity is measured as:

$$\text{AvgServiceTime}(t, n) = \frac{\sum_{i=k-n+1}^{k} \left(T^i_{\text{completion}} - T^i_{\text{arrival}}\right)}{n} \tag{6.4}$$

$$\text{Productivity}(t) = 1 - \frac{\text{AvgServiceTime}(t, n)}{\max\left(T_{th}, \text{AvgServiceTime}(t, n)\right)} \tag{6.5}$$

where

- T^{i}_{arrival} is the time when the i-th order arrives and starts to wait in the dispatching queue.
- $T^{i}_{\text{completion}}$ is the completion time of the i-th order, i.e. when the requested item is transported to the delivery point.
- T_{th} is a threshold value that specifies the largest tolerable service time.

The overall satisfaction of the two goals can be evaluated according to Pareto efficiency or a weighted sum:

$$\text{Overall}(t) = w_s \cdot \text{Safety}(t) + w_p \cdot \text{Productivity}(t) \tag{6.6}$$

where $w_s + w_p = 1; w_s, w_p \in [0, 1]$.

6.6 Experimental Evaluation

The aim of the evaluation in this chapter is to assess the proposed solution by demonstrating the necessity and advantages of the DT *runtime* analysis compared to offline analysis. As stated in Section 6.3: how can the DT better explore the design space, and better understand the rules in dynamic changing contexts or scenarios? The experiment will evaluate DT-enabled analysis in a changing environment and its ability to obtain insights in those scenarios.

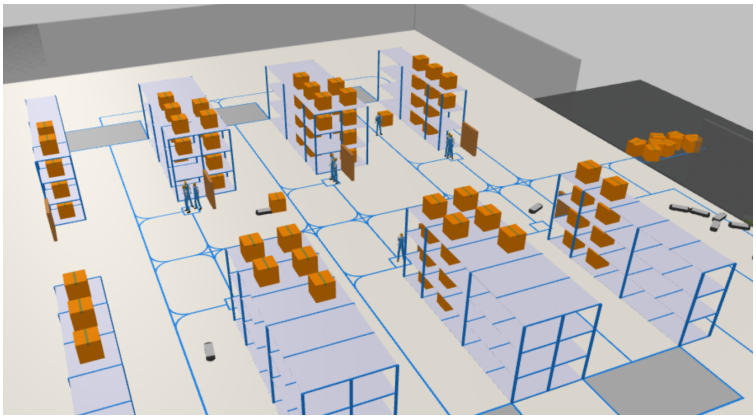


Figure 6.6: The simulation model of the use case.

6.6.1 Experiment Setup

The testbed of all the experiments in this chapter is a MacBook Pro 2015 laptop with macOS 10.15.7, which has a 2.9GHz dual-core Intel Core i5 CPU, 8GB RAM, and 512GB SSD. The DT simulation model of the use case is implemented using AnyLogic 8, as shown in Fig. 6.6. In total 15 AMRs and 9 workers are involved. AMRs and workers both move 1 m/s at maximum. The safety compliance rule embedded in each AMR is: *if there is any obstacle, AMR, or human worker within y meters, reduce the speed to 0.5 m/s*. Here y is a parameter subject to change during runtime with the DT simulation analysis. Collision avoidance of the AMRs is enabled by setting the minimum allowed distance to an object to be 0.5 meters, which ensures a minimum safety requirement. The parameter y provides an extra level of safety.

The environment is designed to be dynamic, with orders arriving at varying rates throughout time. Combined with the intrinsic complexity due to interaction and collaboration between workers and AMRs and the considered constraints for the given context (e.g., safety, spatial distance, etc.), the resulting productivity is significantly impacted.

The experiment in this section considers a *two-phase* scenario to illustrate such dynamism: in *Phase 1*, the orders arrive every 50 seconds, which is considered a relatively

Figure 6.7: Productivity (service time) monitoring for the two-phase scenario with the rule parameter $y = 5$. Phase 2 causes an increase in service time.

slower rate for this case study; later in *Phase 2*, the orders arrive at a much faster rate – every 15 seconds. The two phases depict two typical scenarios. In Phase 1, there is no need for the workers and AMRs to hurry to fulfil the order. However, in Phase 2, time is more stringent and more AMRs may be needed, which can lead to a more crowded warehouse and potentially unsafe conditions for human workers. This can result in a violation of safety compliance measures.

6.6.2 Phase Change

This chapter first shows that a rule leading to acceptable goal satisfaction in one phase may have poor performance in a later phase using a trivial parameter value. The rule parameter y of the AMRs is set to be 5 in both Phase 1 and Phase 2. The simulation result is shown in Fig. 6.7 and Fig. 6.8.

During Phase 1, productivity remains at an acceptable level as the curve remains relatively flat, indicating that no tasks are accumulating. However, in Phase 2, following the same rule parameter results in a steady rise in service time, which indicates that the tasks cannot be completed promptly. Furthermore, regarding the safety concerns depicted

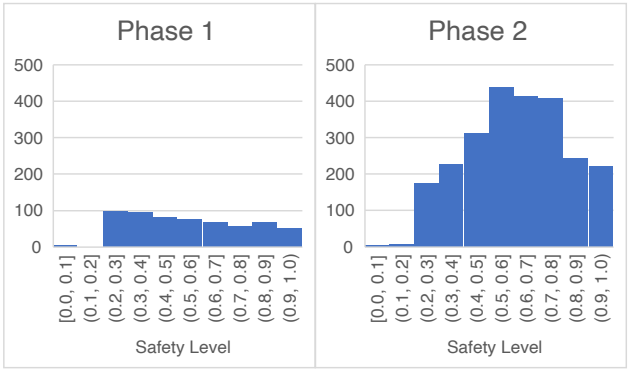


Figure 6.8: Histogram of the monitored overall safety level (measured by $\text{Safety}_{\text{min}}$ in equation (6.3) with threshold $d_{th} = 4$ meters) through time in the two phases when the rule parameter $y = 5.$, excluding the value of 1. The safety level in Phase 2 is more likely to be lower compared to that in Phase 1.

in Fig. 6.8, it is evident that the safety level of workers in Phase 2 is lower than that of Phase 1. This is because workers tend to be closer to AMRs more frequently during Phase 2. Therefore, according to the data, the rule must be adapted in Phase 2 to re-optimise the trade-off between productivity and safety.

It can be argued that such knowledge can be obtained through offline testing during design time. However, before deploying the system to production, it is challenging to predict the occurrence of a phase change and its characteristics. Offline analysis requires identifying all potential phases, and considering all possible world states when the phase change occurs. Due to the immense number of potential combinations of states and phases in such a multi-agent system, this exhaustive offline analysis is not feasible.

Nevertheless, with a DT, compliance and safety engineers can use the DT to conduct *runtime* what-if analysis for productivity vs. safety compliance, and to evaluate different alternative rule designs when a specific phase change is identified during runtime. The premise is that the exercise can help in better impromptu and "directional" planning and optimisation of the production environment specifically for the current state and phase, through adaption and/or tuning the parameters, first tested and evaluated in the simulation

model during runtime, before it gets implemented into the physical system[1].

6.6.3 Runtime What-if Analysis

Then, this thesis intends to demonstrate the advantages of runtime what-if analysis enabled by the DT on how it facilitates examining various design alternatives (parameter value of the rule) during Phases 1 and 2 to obtain insights into their performance. Ideally, the optimal design alternative should maximise both the safety level (safety compliance) and productivity. However, a trade-off analysis between the two goals is usually inevitable as there is a conflict between safety and productivity. Human preferences thus play an important role in prioritising the two goals to select the optimal design alternative at runtime. Notice that using the parameter y adds an extra safety guarantee beyond the basic requirement of the safety compliance standard for the emergency stop. A low safety level is therefore acceptable from the compliance point of view. The experiments below will show how DT can enhance the understanding of candidate solutions with trade-off analysis in a dynamic environment considering different human preferences.

The results of running what-if analysis to predict the performance of different design alternatives for phase 1 and phase 2 are shown in Figure 6.9. To show the impact of human preferences, consider two extreme scenarios: productivity is strictly preferred, or safety is strictly preferred. According to Figure 6.9, if maximising productivity is preferred, the optimal value is 1 for both phases. This is demonstrated in Fig. 6.10, which displays lower service time. Similarly, if safety is preferred in both phases, then in Phase 1, the optimal value for y is 3, but in Phase 2, y should be changed to 4.5. In practice, preference is a human factor and thus is subject to change during the production process. If safety is preferred in

[1]Such analysis requires faster-than-real-time simulation for multi-agent systems, which can be achieved by various execution speedup approaches such as parallel and distributed simulation [226, 153], but this is not the main focus of this thesis.

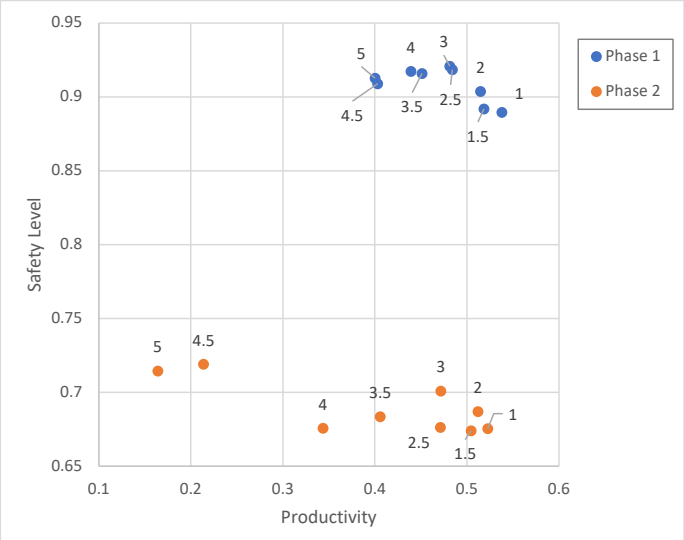


Figure 6.9: The safety and productivity metrics when different design alternatives (the choice of the rule's parameter value) are applied to Phases 1 and 2. Each data point represents the result of selecting a parameter value y. The data labels show the value of y for each data point. The safety level is calculated according to equation (6.3) with $d_{th} = 4$ meters and the productivity is calculated with $T_{th} = 400$ seconds.

Phase 1, but in Phase 2, the productivity needs to be maximised, then y should be first set to be 3 in Phase 1. Then later when the DT detects the environment changes to Phase 2, it needs to run what-if analysis as plotted in Figure 6.9 and changes the y value to 1 according to Figure 6.9. Therefore, with DT runtime analysis, the variable human preferences can be incorporated into the assessment of design alternatives for the trade-off analysis.

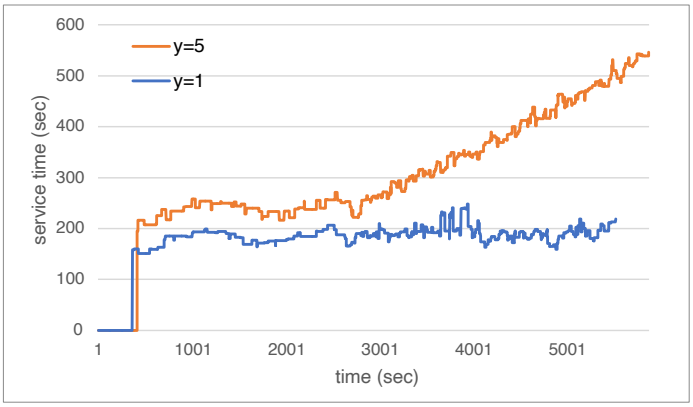


Figure 6.10: Comparison of two design alternatives in both Phase 1 and Phase 2.

Additionally, it can be observed that different phases have varying sensitivity to design

alternatives. Fig. 6.9 clearly indicates that Phase 2 is much more sensitive to parameter values than Phase 1. Consider the comparison between values 3 and 4.5 as an example. During Phase 1, value 3 demonstrates greater Pareto efficiency due to its higher safety level and productivity. However, in Phase 2, both values are on the Pareto front, with value 3 exhibiting higher productivity and value 4.5 demonstrating a higher safety level. This serves as a reminder that parameters that prove optimal in certain phases or scenarios may not necessarily remain optimal in the future. Therefore, it is imperative to understand that during Phase 2, proper adaptation of the rule parameter is crucial, as even the slightest differences in values can significantly impact productivity. Efficient acquisition of such knowledge is only viable with a DT that conducts runtime analysis for the specific phase and state observed at runtime, as opposed to exhaustive offline testing of all possible phases and world states.

6.6.4 Evaluation Summary

The above analysis shows that dynamic phase changes in the environment require re-evaluations of design alternatives to adapt for better trade-offs between compliance and quality goals. However, such analysis is infeasible offline because of the unpredictable nature of the complex system and the resulting vast exploration space. It is difficult to predict when and what specific phase can occur under what world state; thus, offline analysis can only exhaustively explore all possibilities.

In contrast, the DT can explore alternative designs more efficiently by obtaining knowledge specific to the phase change that is happening in production. This results in a targeted exploration without simulating scenarios unrelated to the monitored reality. The DT also supports runtime *dynamic* preference, an unpredictable human factor that is difficult to handle by offline analysis. All the above capabilities are made possible through the DT's monitoring ability and the multi-model approach with high-fidelity simulation models and

goal models.

Compliance and safety engineers can use the approach in online adaption, where the physical system and the DT embark on frequent and continuous symbiotic updates, where the physical system is updated with the optimal solution for a given context, after being tested in the DT.

6.7 Summary

This chapter proposes a novel goal-aware Digital Twin (DT) approach for runtime compliance management, leveraging multi-agent simulation and goal modelling. The approach is demonstrated using a smart warehouse human-robot collaboration case with stringent safety compliance and productivity goals. Alternative rule designs of each agent are first elicited with GRL goal modelling for safety compliance and productivity, along with other dependencies. Then these alternatives will be evaluated at runtime at the specific scenario encountered with the support of faster-than-real-time high-fidelity agent-based simulation. Finally, the rule leading to the "best" trade-off will be applied to the agent on the robot. Through Pareto efficiency, the trade-off between compliance and productivity can be evaluated. The proposed DT-oriented architecture provides compliance engineers with forecasting capability for refining and switching between the different rules before enacting the real system at runtime. The experimental evaluation demonstrated the need for runtime adaptive compliance and the efficacy of using the DT's what-if simulation analysis for adaptive control. It is shown that a DT can more efficiently explore the design space and understand the rules in different contexts. Future work will examine how the vision can be extended to provide cognitive and proactive adaptive control considering various compliance requirements, context and time-varying trade-offs.

Chapter Seven

Architecting Explainable Human-In-The-Loop Digital Twins

A self-aware DT is essentially data-driven, where data and accumulated knowledge steer different levels of awareness. While the previously proposed generic reference architecture and reference model have provided a blueprint for *self-aware DTs of self-aware systems*, they did not consider the role of human input in runtime decision-making. With human-in-the-loop, humans may further enhance the decision-making of the DT, and the DT can also inform humans with new knowledge of the world. It is thus crucial for humans to comprehend the reasoning behind the DT's decisions to foster trustworthiness, in order to decide for possible intervention or take-over. Trust and comprehension of the DT's decisions drive the need to incorporate explainability into the DT. This chapter enriches the generic reference architecture by incorporating DDDAS principles to delineate the data-driven decision-making processes and provide explainability for various types of autonomous decision-making. This chapter instantiates the architecture to support certain levels of awareness, specifically goal-awareness with explainability.

7.1 Overview

This chapter specifically focuses on applying the DDDAS principles to the detailed design and modelling of the feedback loop of the DT-based system. This chapter particularly discusses how the knowledge management flow should be addressed in the feedback loops, and what the role of knowledge is for the intelligent DT of intelligent systems.

Digital Twins (DTs) are data-driven models that serve as real-time symbiotic "virtual replicas" of real-world systems. Digital Twin modelling can leverage principles of Dynamic Data-Driven Applications Systems (DDDAS) in its realisation and refinement. The DT feedback loops, for example, can be designed following the DDDAS paradigm to continuously steer the measurement, modelling and analysis [34]. As is the case in some DDDAS applications, some of the DT-related decisions cannot be fully autonomous and need to be enacted by human operators [126, 127, 125]. There can be decisions that are counter-intuitive, which may confuse the human operators. Humans need to understand the patterns or rationale behind the system's decision-making logic in order to trust and follow the instructions or make further decisions. Providing explanations to humans (e.g., operators, developers or users) can facilitate their understanding of the rationale behind the decisions and their rightfulness for a given context. Explanations can also provide assurance for humans to trust the autonomous adaptation by the DT and its underlying DDDAS. Humans can also examine and inspect the operation of DDDAS. In case of any anomaly or fault identified from the explanations, humans can intervene in the system to assist or improve its decision-making.

The chapter makes the following novel contribution: (1) It motivates the need for providing explainability in human-in-the-loop Digital Twins, leveraging DDDAS principles and its design for feedback loops; (2) As a prerequisite for explanation is the identification of areas within the system, where the provision of explanation is crucial for trustworthy service; the chapter describes a data-centric strategy to identify areas within DDDAS and

DT that require explanations; (3) It provides an enriched reference architecture model for the DT, building on DDDAS and supported with explainability; the model extends the authors' previous work on digitally Twinning for intelligent systems and illustrates the utility gained for supporting explainability; (4) It discusses the trade-offs and costs of providing explainability.

7.2 Background and Related Work

In recent years, there has been increasing interest in the explainability and interpretability of Artificial Intelligence (AI), denoted as XAI (eXplainable Artificial Intelligence) [172]. The empirical success of deep learning models leads to the rise of opaque decision systems such as Deep Neural Networks, which are complex black-box models. Explanations supporting the output of a model are crucial in providing transparency of predictions for various AI stakeholders [22]. Explanation relates to how intelligent systems explain their decisions. An Explanation refers to causes, and is an answer to a *why-question* [172].

In Digital Twins, decisions can be a product of autonomous reasoning processes and require explanation. The decision model of a DT may suggest locally sub-optimal decisions but also has the risk of making mistakes. A typical example is DTs that employ reinforcement learning, a sequential decision-making model. Reinforcement learning aims at deciding a sequence of actions that maximises long-term cumulative rewards. However, an optimal action in the long term may, at present, be sub-optimal or even temporally lead to undesired system states. The lack of explanation for sub-optimal actions can lead to confusion and difficulty for humans to distinguish whether the autonomous system has made a mistake or the "faulty" decision is intentional for the long-term benefit. With explanations, human operators and stakeholders can understand the rationale behind the decisions, and increase

trust in the DDDAS and DT-based systems they utilise for decision support.

Some existing preliminary works have discussed explainability in dynamic data-driven Digital Twins. One way is to use white box models based on physical or mechanical principles so that the system is interpretable [239]. For black box models such as deep learning, XAI techniques are powerful techniques for the transparency of prediction or decision-making. IoT devices can facilitate the monitoring and data collecting of the DT at runtime [29]. In [119], explanations are provided for dynamically selecting a Digital Twin model from a library of physics-based reduced-order models. In this work, based on sensor measurements, a trained optimal classification tree matches a physics-based model that represents the structural damage state at the moment of measurement. The approach is demonstrated in a case study of unmanned aerial vehicles (UAVs). The work in [181] assumes DT utilises deep learning models to perform time-series forecasts for anomaly detection. A three-layered architectural pipeline is proposed to generate explanations of DT's prediction. In [169], the authors integrate self-explainability techniques with digital twins. The explainability of a DT is enriched using the MAB-EX framework, a variation of the MAPE loop. The work further suggests that the explanation model should be built from formal system description models, process models, and reasoning models.

Apart from explainability at runtime, the design process of a system can also benefit from explanations. In [246], the authors propose a framework for the explanation needed during the modelling process, to support the collaboration between modellers and stakeholders for the clear and explicit understanding of the modelling process.

Despite the work in enriching DTs with explainability (see Table 7.1), research addressing the explainability of *decisions* autonomously made by the DT is still at the preliminary stage [119, 169]. There is a need for systematically synthesising and classifying the specific types of decisions that explainability can support. Additionally, although ex-

plainability implicitly relates to human-in-the-loop and trust, the role of humans and their interaction with different components and runtime internal processes of the DT-based system is not adequately identified and classified. This chapter combines DDDAS principles with DT and classifies from the DDDAS perspective the different roles of explainability supporting data-driven decisions and possible challenges. This chapter further identifies how explainability relates to humans-in-the-loop specifically in how humans and the DT can mutually benefit from explainability from a system architecture perspective.

Table 7.1: Related work for explainability and DT.

Reference	**Domain**	**Phase**	**What to explain**	**Purpose**
[246]	Manufacturing	Design process	Design and modelling	Clear and explicit understanding of the modelling process
[181]	Anomaly detection	Runtime	Predictions	Explain the time-series forecast of deep learning models
[12]	Soil carbon emission prediction	Runtime	Predictions	Justify the reliability of predictions by the DT and possibly inform model retraining
[132]	Prognostics and health management	Runtime	Predictions	Explain the prediction of remaining useful life
[169]	Cyber-physical systems	Runtime	Autonomous behaviours	Explain DT's decisions
[119]	Unmanned aerial vehicles	Runtime	Autonomous decisions	Model selection
This thesis	Cyber-physical-social environment	Runtime	Autonomous decisions	Explain DT's **different types** of decisions

7.3 Motivating Example

In this section, a motivating example similar to the one in Chapter 6 is presented. The example assumes a Digital Twin of an intelligent system. The intelligent system is composed of multiple collaborating computing nodes, each of which is controlled by an onboard intelligent agent. Each agent learns by accumulating knowledge of its environment and interaction with other agents. Each agent acts autonomously based on its knowledge, but its quality of decision-making may be restricted by the processing capability of the node or the power consumption if it is battery-powered. Therefore, the agents can act autonomously most of the time, but may need a Digital Twin of the entire system of agents for global optimisation of trade-off analysis between different goals and requirements. The resultant control decisions can be in the form of changing the decision algorithms/models of the agents (as mentioned in Section 3.3.1,or modifying the parameters [166]. Smart warehouse is an application domain where the above-mentioned system can be applied. Multiple Autonomous Mobile Robots (AMRs) can travel within the warehouse for collaborative picking-up and delivery tasks. Each AMR is controlled by an agent that decides which task to take and which other AMRs to collaborate with. Each agent uses rules to decide its actions, but the optimality of rules can be contextual. Rules need to be adapted in order to reach optimal performance in different runtime contexts. However, it is risky for agents to self-adapt the rules, since each agent only has partial knowledge about a subset of the entire environment in space and time. The quality of self-adaptation is threatened by the lack of knowledge of other parts of the environment. A Digital Twin of the warehouse is able to provide more informed decisions on modifying the rules of agents, with a global view of the entire system and what-if predictions of various possible scenarios.

The Digital Twin utilises the DDDAS feedback loop to dynamically ensure both the state and knowledge of the intelligent system are accurately modelled (see Chapter 5). Based

on the model, predictions of various what-if scenarios can then further inform decision-making to optimise the behaviour of the system. Due to the nature of autonomy and high complexity within the system, decisions made by the DT may still be imperfect or even faulty. Human supervision and intervention are necessary, which requires explanations from the DT for humans to identify potentially undesired decisions.

7.4 A Reference Architecture for Explainable Digital Twins Leveraging DDDAS Principles

To address the issue of explainability in the motivating example, this chapter presents a novel reference architecture for an explainable Digital Twin system, leveraging DDDAS feedback loops in Fig. 7.1. The architecture extends on the holistic architecture in Section 3.3 (Figure 3.4) to provide refinements of the architecture informed by DDDAS feedback loop design and enrichment with primitives for data-driven explainability. The architecture utilises interpretable machine learning and goal-oriented requirement modelling to provide explanations of decisions that are driven or influenced by data. The proposed architecture adopts the classical design of DDDAS computational feedback loops: (1) *sensor reconfiguration*, which guides the sensor to enhance the information content of the collected data, and (2) *data assimilation*, which uses the sensor data error to improve the accuracy of the simulation model [34]. Additionally, this chapter refines the above two loops with human-in-the-loop inputs, as explainability is essentially a human-centric concern. Utilising the simulation model and analysis, the system behaviour can be dynamically controlled in a feedback manner.

In Fig. 7.1, the physical space at the bottom contains the system that is composed of intelligent agents, as in the motivating example. The Digital Twin contains a simulation model that replicates the physical space. The agents in the model are identical software

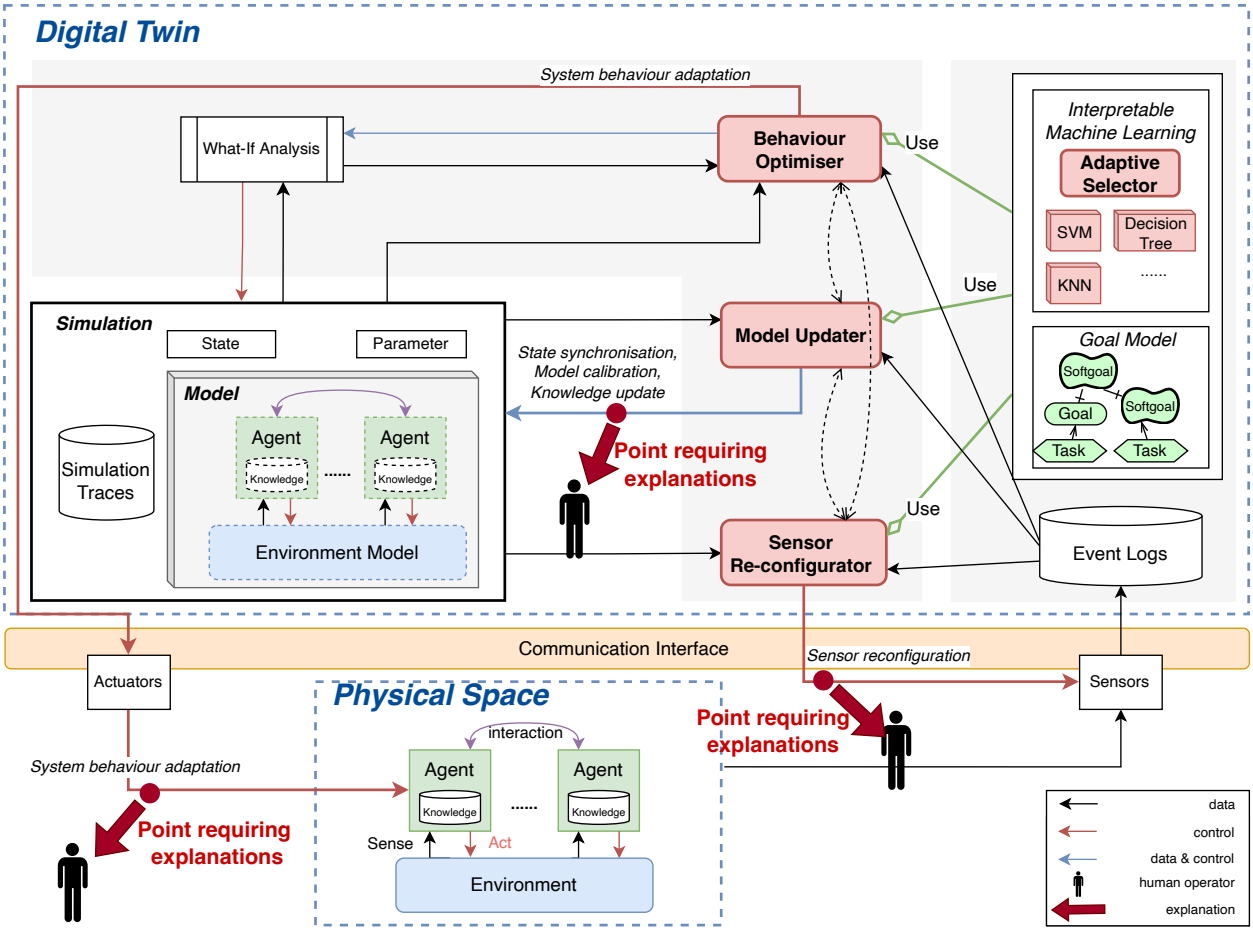


Figure 7.1: A novel reference architecture for explainable human-in-the-loop DDDAS-inspired Digital Twins.

programs to the agents in the physical space.

There are three controllers that utilise feedback loops for autonomous decision-making: *Sensor Re-configurator*, *Model Updater*, and *Behaviour Optimiser*. Each controller incorporates interpretable machine learning and goal models to provide explanations to human operators. The three control components can also interact with each other. For instance, the *Behaviour Optimiser* may lack high fidelity data to justify and validate its decision, then it can notify *Sensor Re-configurator* asking for further enhanced sensory data.

7.5 Explainable Decisions for Human-In-The-Loop Digital Twins

Human factors can be present in the two computational feedback loops in the DDDAS-based DT and can be captured in an additional physical human-in-the-loop feedback process, as illustrated in Fig. 7.1. Explanations are needed to enhance the trust of humans in the system and/or allow humans to find anomalies or faults in the autonomous process in order to improve the decisions made by the system. With explanations provided, humans can trace the input data and how the data are assimilated to generate the adaptation decision. Inadequate decisions can be identified and their cause can be located: because of faulty input data or during the reasoning process.

Explanations are closely linked to decisions steered by data, accumulated knowledge or computations by data assimilated into the system. The proposed architecture aims at providing explanations on how and why decisions or adaptations are made, linked to data: the data that support a decision need to be traced, since the decision is a result of data input and how the data is then processed and/or manipulated. In the architecture, the adaptations of the three controllers represent three categories of decisions that can be made by the DDDAS-based Digital Twin: (1) *measurement adaptation*, (2) *model adaptation*, and (3) *system behaviour adaptation*. For each type of adaptation decision, explanations are required to increase the transparency of decisions for humans (shown as "Point requiring explanations" in Fig. 7.1). Goal models help in presenting the satisfaction of system requirements by their utility. Each controller uses interpretable machine learning for its decision-making to provide explanations directly.

7.5.1 Explanation for Measurement Adaptation

Measurement adaptation is a part of the sensor reconfiguration feedback loop, which controls what and how data are collected. This type of adaptation is decided by the *Sensor Reconfigurator* in the architecture. Explanations are essential to answer: what is the purpose of a measurement adaptation, and how is it beneficial? Examples of reasons can be to estimate the state more accurately, system behaviour adaptation requiring more data for validation, or cost-effective concerns such as energy.

Different aspects of adaptation on the measurement are as follows. Each adaptation requires explanations.

- **Data source**: The Digital Twin can dynamically decide what sources of data to be collected. In the motivating example, knowledge and state are two sources of data. When identifying discrepancies between the model and the real system, comparing knowledge between the two can lead to more cost-efficient results than comparing states in certain scenarios (see Chapter 5).
- **Sampling rate**: The frequency of sensing can be adjusted. A sensor can be put into sleep mode for the reason of energy saving if its readings can be predicted by historical data or other data sources [173].
- **Sensing fidelity**: Sensors can send back imagery data in high or low resolution, based on the trade-off between data enhancement due to higher fidelity observation vs. cost [34].
- **Sensor placement**: The locations of sensors can be adjusted to monitor areas of interest, possibly with multiple sensors from different perspectives.
- **Human monitoring**: Humans can follow instructions from the Digital Twin to mon-

itor and inspect the system and environment on the spot and report data back.

7.5.2 Explanation for Model Adaptation

The aim of model adaptation is to increase the fidelity of the simulation model. This adaptation decision is made by the *Model Updater* in Fig. 7.1. Explanations should justify why the adaptations are beneficial in improving fidelity. The adaptation can be state replication and estimation, parameter calibration, and knowledge update.

Data assimilation is a common approach for state estimation, which aims at finding a "best" estimate of the (non-observable) state of the system. However, the motivating example is different from the conventional application scenarios of data assimilation, such as the Kalman Filter. The system is discrete-event and involves complex interactions between different entities, which cannot be easily characterised by mathematical properties and assumptions with numerical models. Although a data assimilation framework for discrete event simulation models has been proposed [105], the example system itself is driven by inherent computation and knowledge of the agents, which poses an additional challenge for the model to replicate not only the state but also the knowledge of the system.

Some research efforts have already been made to provide explanations for model adaptation. Interpretable machine learning can be used to select within a library of models based on real-time observation [119]. Model adaptation can be driven by the discrepancies between sensor data and the data from the Digital Twin model. In [85], a Gaussian Mixture Model-based discrepancy detector is first designed to identify anomalies. Then the detected anomalies are further classified into different types by Hidden Markov Model. The classification of anomalies can serve as the explanation for discrepancies to support later decisions. However, their work does not involve feedback loops that correct the anomalies

after classification.

7.5.3 Explanation for System Behaviour Adaptation

System behaviour adaptation refers to how the Digital Twin model is used for simulating different what-if scenarios to inform decisions on optimising the system behaviour. This type of decision is made by the *Behaviour Optimiser* in Fig. 7.1.

What-if analysis can be self-explainable, since *what if*-questions are essentially asking explanatory questions of *how*, and are just a contrast case analysing what would happen under a different situation or with an alternative choice [172]. Nevertheless, the process of exploring all different what-if scenarios can be expensive thus requiring to be optimised by e.g. searching algorithms.

Explanations can also be achieved by providing the satisfaction of the system requirements during operation [254]. These requirements are imposed by experts during the design time of the Digital Twin system, which can involve the task goals (e.g. delivery efficiency in the warehouse example) or system performance (e.g. computational load of the Digital Twin). Goal models can be utilised to provide explanations [254], which is also incorporated in the proposed architecture.

7.6 Discussion

7.6.1 Trade-off Analysis

The proposed architecture has taken a data-driven stance to explainability of decisions in the system. However, dynamism in data sensing, assimilation and processing can make unneces-

sary explainability costly. Although explanation is beneficial for transparency, trustworthiness and understandability, the trade-off between performance and explainability needs to be considered in dynamic and data-driven systems. Additionally, decision-making algorithms and models differ in the quality of decisions; henceforth, the levels of explanation needed [22] can vary with the quality of decisions induced by these models. The empirical evidence that evaluates how end users perceive the explainability of AI algorithms has been established [100]. The trade-off is situational and depends on the complexity of the problem. Decision models may rank differently in performance vs. explainability when solving different problems, along with the varying complexity of input and training data (e.g. tabular data or images) [100]. For highly dynamic data-driven open systems, the data sensing can vary in complexity, quantity and refresh rate. The type and sophistication of the decisions can also vary. There is a need for the DT to self-adaptively decide when to use which decision models and the level of explanation required. In the reference architecture in Fig. 7.1, an *Adaptive Selector* is proposed to switch between the models in different runtime situations dynamically. One possible solution to adaptive selection is to abstract the system into different states. Each state can be pre-calculated for its requirement on the level of explanation. The *Adaptive Selector* monitors the state to make selections.

The choice of decision models may further influence the trust of human operators and their willingness to enact the decisions. The DT's dynamic trade-off analysis demands awareness of the cost due to the explanation's impact on human behaviours. An adaptive cost-aware approach has been proposed in [147], which uses probabilistic model checking to reason about when to provide explanations in a case study of web services. This study assumes that explanation can improve humans' probability of successfully performing a task, but will incur delayed human actions, because humans need time to comprehend the explanation.

7.6.2 Evaluation Framework

The proposed architecture can be evaluated from two perspectives: 1) evaluating the design trade-offs at an abstract level and 2) instantiating the architecture into prototype systems and evaluating it with simulation-based experiments. This section provides an evaluation framework to guide future studies in evaluating the added value of explainability for DT-based systems from the above two perspectives.

At a more abstract level, the inclusion of explainability can be viewed as an architecture design decision. Well-established architectural evaluation methods may be used to evaluate the extent to which explainability can improve the utility of the architecture under design, against other quality attributes of interest that need to be satisfied within the architecture. Scenario-based evaluation methods, such as ATAM [124], may be used during the inception and development stages of the architecture to inform the design of explainability. Scenarios are human-centric and context-dependent. Types of scenarios can be typical, exploratory, or stress scenarios. The extent to which the design for explainability is fit depends on the choice of scenarios. Scenarios may focus on issues such as the need, effectiveness and added value of having explanation embedded as part of the architecture. Examples of the added value of supporting explainability in the data-driven cycle include: ensuring compliance (safety, security, etc.) and avoiding accidental violation of compliance as data is streamed, analysed, processed and/or decisions are actuated; tracing back the cause of issues attributed to data and decisions, mainly for debugging and forensic investigation; transparency of decisions and control by explaining data that can drive control; explaining the risk of critical data-driven decisions, among the others.

Considering the *Updater* as an example, the parameter calibration can be triggered by policies and the measured discrepancy between the simulation and the real system. Calibration should be triggered only when necessary, since estimating the parameters requires

excessive computational resources. However, issues such as sensor anomaly may trigger the calibration by mistake, causing unnecessary computational costs. Explanations can inform humans to trace the need for calibration and identify such a measurement anomaly. Frequent explanations contribute to cautious anomaly identification by always involving humans, but can instead cause annoyance and decreased quality of experience (QoE) of the humans when even trivial decisions need confirmation from humans. The trade-off analysis between computational cost and QoE is essential in evaluation.

Regarding the evaluation framework from the instantiation perspective, this chapter proposes the following methodology to evaluate the added value of explainability with controlled experiments. The methodology provides actionable directions for evaluating the benefit of explainability added to the generic architecture. The implementation is regarded as a future work to be conducted beyond the current content of the thesis. In general, the evaluation objective is to assess how the explainability-enriched architecture proposed in this chapter can improve the autonomous decision-making of the DT-based system, and enhance the user trust in the system. The hypothesis is that with explainability, the decision-making of the DT-based system can be improved by reducing sub-optimal or harmful decisions generated through its autonomous reasoning process, such that the system performance is improved. Additionally, with explanations provided by the DT-based system, human trust in the system's autonomous decisions should be higher than with no explanations.

To instantiate the proposed architecture in this chapter, the evaluation framework will enrich the mechanisms proposed in Chapters 6 and 5 with explainability. Specifically, in the autonomous knowledge equivalence maintaining and runtime goal-aware what-if analysis proposed in Chapters 5 and 6, the evaluation framework will adopt XAI techniques such as optimal classification tree and SHAP [157] to enable explainable decision-making. Human users should be involved in examining the decisions generated by the DT-based system regarding P2V model update and V2P adaptation. If the human user identifies sub-optimal

decisions, they should intervene in the autonomous process and manually apply the decision that they regard optimal.

The evaluation should regard the DT-based system that does not provide explanations as a baseline, and compare with the same DT-based system but with explanations. The evaluation metrics will quantify the added benefit of providing explanations from two perspectives: 1) the performance of the real-world system quantified by utility functions, as adopted in Chapters 5 and 6; 2) the trust of human users in the autonomous decision-making process of the DT [45, 123]. To quantify the trust, in-experiment rating, post-experiment questionnaires and interviews with human participants are needed to measure the trust level [130, 267]. The evaluation framework will adopt the same application scenarios as the work in Chapters 6 and 5 with smart logistics and smart drone surveillance. The simulation model will be implemented using AnyLogic and Repast by extending the implementation in Chapters 6 and 5 with XAI techniques mentioned above.

Challenges may arise during the evaluation and implementation process. One significant challenge involves creating user-friendly explanations that cater to individuals with varying levels of knowledge. While large language models can serve as a viable approach for generating natural language explanations, they do face issues related to hallucination, which can lead to false or misleading information [106]. Additionally, it is essential to provide explanations at an appropriate level of detail based on the background knowledge of users. Designing the metrics of trust is another challenge. Although [130] provides some examples of metric designs for questionnaires, these metrics should be customised and adapted to the specific application context of explainability and DTs. If XAI is employed, another challenge emerges: obtaining an appropriate training dataset. Either an open dataset is needed, or the dataset should be generated through simulation. Lastly, runtime analysis poses further challenges in fast simulation execution, which can be solved by using parallel and distributed techniques [82].

7.6.3 DDDAS vs. Self-Aware Digital Twins

Though the coordination between the physical system and the digital twin can exhibit some similarities to that of classical DDDAS, there are several distinctions: (i) the coordination operates on finer dimensions of knowledge that can relate to stimuli-, goal-, time-, and interaction- awareness; (ii) the digital twin is not restricted to real time simulation and symbiotic back and forth updates between the two worlds but also includes additional cognitive abilities that can make the digital twin self-aware and consequently provide more intelligent updates to the physical world; (iii) the digital twin can also act as Cyber foraging environment, acting as "surrogate" for real-time processing and sophisticated analysis, planning and simulation that would be computationally expensive and extremely difficult to carry out in the drones network due to their limited computational power; (iv) the knowledge can be either sensed through data assimilated from the drones in physical system and then analysed in the digital twin as "surrogate" or generated through simulation to anticipate for eventualities and stress scenarios that would be difficult to track and trace in real settings. Knowledge is an enabler for different levels of self-awareness in digital twins, with intelligent decisions fed to the real system.

7.7 Summary

This chapter has aspired to investigate the problem of explainability in human-in-the-loop dynamic data-driven Digital Twin systems. This chapter has contributed to a novel reference architecture of the DT, which draws inspiration from the design of DDDAS feedback loops and enriches it with primitives for explainability. The architecture is designed to identify and elaborate on where and when explanations can support decisions that are data-driven or influenced by data assimilated and/or computed by the system.

The areas that need explanations are identified by focusing on three types of adaptation decisions, traced to data. This chapter has refined and enriched the architecture to provide explainability linked to the decisions of three controllers. The controllers use interpretable machine learning and goal models to provide explanations. Future work will focus on further refinements and implementation of variants of explainable Digital Twin architectures for autonomous systems. This thesis will examine cases from smart warehouse logistics and manufacturing, where humans and machines collaborate to inform the design and evaluate the effectiveness of adaptive explanations. Special attention will be paid to the impact of explanation on the behaviours of humans. The *Adaptive Selector* in the architecture will be instantiated by algorithms that use state-based approaches to dynamically switch the decision-making models.

Chapter Eight

Conclusion and Reflection

8.1 How the Research Questions Have Been Addressed

As mentioned in Chapter 1, the aim is to investigate the symbiotic relationship between intelligence and Digital Twins and how to engineer mutual intelligence enrichment between the DT and the real-world system. This thesis is motivated by the inadequacy of the research in a novel problem scenario where both the real-world system and its DT have online knowledge management capabilities. To address the gap in characterising the interaction between the two knowledge management processes and their mutual benefits, this thesis has proposed reference architectures and detailed mechanisms for addressing various dimensions of mutual intelligence enrichment. This section summarises the answers to the research questions that drive this research.

- **RQ1: How to design an intelligent DT that combines knowledge management and simulation what-if analysis to assist self-aware systems?** The thesis has reviewed related literature in Chapter 3 and 4, to identify the engineering efforts in the intelligence of the DT and the intelligence of the real-world system. The literature review result reveals the inadequacy of digitally twinning intelligent systems that self-

manage their runtime knowledge. In Chapter 3, a generic reference architecture has been proposed for intelligent DTs of self-aware systems and the corresponding formal description. There is also a lack of effort in engineering the cognitive capabilities of the DT to combine prescriptive simulation analysis, self-*, and dynamic knowledge management/learning. Chapter 4 leverages design principles of self-awareness to propose a refined reference model for the DT to address all three capabilities.

- **RQ2: How to achieve mutual intelligence enrichment between the intelligent DT and the self-aware systems?** Mutual intelligence enrichment is approached by examining specific modules and processes of the proposed generic reference architecture. This thesis has proposed refinement of the generic reference architecture and the design of the mechanisms in P2V and V2P directions of mutual enrichment, respectively. Mutual intelligence enrichment is achieved by utilising the runtime knowledge from the DT and that of the real-world self-aware systems.

  From the P2V perspective, the review in Chapter 5 demonstrates the need for novel equivalence checking and maintaining approaches when digitally twinning intelligent systems. Most of the existing approaches do not study the scenario where the physical twin is characterised as multiple intelligent agents that have online knowledge management capabilities. The existing works in checking the equivalence between the DT and the real system are limited to the comparison of states, input or output, with no focus on knowledge. Therefore, Chapter 5 proposes the novel notion of knowledge equivalence, requiring runtime knowledge of the self-aware system to be replicated to the DT in real-time. A novel metric based on knowledge discrepancy is also proposed to guide the update of the DT in a cost-efficient manner. With the quantitative evaluation using a smart mobile camera example, keeping knowledge equivalent and continuously updated is shown to be an inevitable mechanism to ensure the validity of the simulation prediction by the DT. This thesis has also proposed an updating scheme based on

continuous knowledge equivalence checking, which answers the question of when to update the DT's model. The runtime knowledge maintained by the real-world self-aware systems can aid efficient updates of the DT's simulation model. Evaluation has shown that knowledge equivalence checking can achieve more Pareto-efficient performance for cost-efficient model updates.

From the V2P perspective, Chapter 6 leverages both the generic reference architecture (Chapter 3) and the reference model for self-aware DTs (Chapter 4), to investigate how a self-aware DT can be applied to provide insights for the knowledge evolution of a self-aware physical system. A problem domain of runtime adaptive compliance is adopted to demonstrate the applicability of the architecture design. As reviewed in Chapter 6, existing runtime adaptive compliance solutions lack the ability of prescriptive simulation analysis and are not designed for analysing performance trade-offs for complex self-aware systems. Chapter 6 thus proposes a novel DT-based goal-aware self-adaptation approach for rule-based stimulus-aware systems to provide insights for online adaptive compliance. The approach enables design choices elicited from the compliance documents at design time to be computationally analysed by online DT simulation during runtime. In particular, knowledge of the DT, i.e. goals and insights obtained from online simulation analysis, can guide the adaptation of the rules of the stimulus-aware systems. The applicability and necessity of the approach have been demonstrated for adaptive compliance in a smart warehouse logistics example. This serves as a feasibility study to guide future application of the proposed generic architecture to other problem domains.

- **RQ3: If humans are involved in the mutual intelligence enrichment process, how can DTs provide explainability to humans for improved decision-making?** As reviewed by Chapter 7, there is a need to systematically enrich DT's capability for providing explanations for its autonomous decisions. Existing approaches

mostly address the explainability of DTs for predictions rather than the autonomous decision-making process. In Chapter 7, the generic reference architecture is enriched by incorporating explainability into a DDDAS-based DT. The architecture makes a novel contribution in identifying three different types of explainable decision-making processes from a DDDAS perspective: explainable measurement adaptation, explainable model adaptation, and explainable system behaviour adaptation. Chapter 7 has further discussed possible solutions to support the three types of explainable decision-making feedback loops and their synergy with human-in-the-loop. The resultant refined architecture can serve as a blueprint to guide the design of a human-in-the-loop intelligent DT.

8.2 Future Directions

Several future directions of the thesis can be investigated to extend the research horizon for further consolidation of mutual intelligence enrichment.

8.2.1 Other Levels of Self-Aware Digital Twins

In this thesis, mainly goal-awareness is instantiated in the self-aware DT reference model to facilitate the enrichment of the real-world system. More consolidated designs based on other levels of self-awareness of the DT can be further explored. Further formulation on how these levels of self-awareness can support different modules and processes in the DT is still an open problem. Meta-self-awareness, which is the highest level of self-awareness capability to reason for other lower levels of awareness, is not discussed in this thesis. Meta-self-awareness can enrich the DT with a more in-depth understanding of its own reasoning thus boosting more informed self-adaptation of the internal mechanisms of the DT itself. Future research

is needed to delineate the potential for a DT that is meta-self-aware.

8.2.2 Adaptation for Other Levels of Self-Aware Systems

This thesis assumes the real-world system is self-aware. However, the P2V equivalence only studied interaction-awareness and time-awareness; the V2P adaptation mechanisms only examined stimulus-awareness. Further research is needed to involve intelligent real-world systems that are characterised by other levels of self-awareness in the P2V and V2P processes. More specifically, Figure 5.1 has already provided a reference architecture for maintaining equivalence considering other levels of self-awareness in the real-world system. This can serve as a vision to promote future research in P2V knowledge equivalence and how the modules coordinate with each other.

8.2.3 Decentralisation and Distribution

This thesis regards the DT as a centralised entity and implicitly models the entire physical world into one model. There can be various improvements regarding the decentralisation of the DT. First of all, the modelling and replication can be made decentralised. The lifecycle of each digital twin of the real-world entity can be managed independently. As depicted by Figure 5.2, decentralisation for equivalence management can be beneficial since the updating frequency may be different for different real-world entities. An entity whose state and knowledge do not change during a long period of time will need less frequent updates compared to entities that change rapidly. Decentralised equivalence management enables asynchronous updates for different entities, thus reducing unnecessary data transmission and computation. One of the challenges can be: if simulation analysis from a holistic view is needed to simulate all entities altogether, it is crucial to keep all digital twins synchronised. Secondly, the mod-

ules of the DT can be decentralised and distributed. Each module can be engineered to be microservices. Such a distributed approach can result in scalability improvement. However, managing the complexity of distributed modules can also be a challenge.

Distribution of the real-world system may involve heterogeneous sub-systems. Each agent in the real world may possess different levels of self-awareness or different decision models from other agents. In the current model, agents are homogeneous in their decision models. Future work can consider scenarios with heterogeneous agents and how the P2V and V2P can be realised in a centralised or decentralised fashion.

8.2.4 Explainability Driven Analysis

This thesis provides a reference architecture that identifies where explainability is needed. Future work is needed to instantiate the architecture into actionable mechanisms to support decision-making and the mutual understanding between the human and the DT. It is essential to study how the explainability of the DT can affect human intervention and the quality of the final decisions. In addition, value alignment is needed to ensure the objectives and intentions are properly coordinated between the human and the DT [267].

8.3 Closing Remarks

This thesis makes novel contributions to the integration between intelligence and Digital Twins with engineering approaches to enable mutual intelligence enrichment between the DT and the real-world system from a runtime knowledge point of view. This thesis has envisioned the novel notion of *intelligent Digital Twins of intelligent systems* from a perspective where intelligence is discussed in the context of runtime knowledge related to self-awareness.

Implementing intelligence in both the DT and the real-world system can be a promising paradigm that best exploits the computation capabilities of both the real-world system and the DT. Mutual intelligence enrichment can serve as a crucial characteristic towards more cost-efficient and autonomous behaviours of both the physical twin and the digital twin. This thesis envisions that the effort of this thesis on mutual intelligence enrichment mechanism can stimulate further research for various other aspects of *intelligent Digital Twins of intelligent systems*.

Appendix One

Agent Behaviours in the Motivating Example of Chapter 5

Algorithm 5: Drone Agent Behaviour.

```
Data: Knowledge of interaction graph, drone sensing range range, list of all
      received messages msgList,
for every time step do
    /* SENSE                                                              */
    objList ← sense all objects within range;
    imptObjList ← filter important objects from objList;
    msgList.append(messages received at previous time step);
    Clear outdated messages in msgList;
    /* THINK and ACT                                                      */
    if !imptObjList.isEmpty() then
        obj ← randomly select from imptObjList;
        if !obj.isKCovered() then
            // notify
            notifiedDrones ← the top (k − 1) drones ranked by the strengths of
              their links with this drone, which are recorded as the edge weights in
              graph;
            foreach drone ∈ notifiedDrones do
                Send message to drone;
            end
        end
        // follow
        Move towards obj by 1 unit;
    else if !msgList.isEmpty() then
        // respond-and-follow
        Sort messages in msgList in descending order first by the weight of edge
          between the sender drone and this drone, then by the receiving time;
        msg ← the message at the highest rank after sorting;
        obj ← the object advertised in msg;
        Move towards obj by 2 unit;
    else // random walk
        Move towards a random angle by 5 units;
    end
    /* UPDATE KNOWLEDGE                                                   */
    foreach edge ∈ graph.allEdges do
        edge.weight ← γ · edge.weight;
    end
    foreach obj ∈ ObjList do
        foreach d ∈ all surrounding drones do
            if d.isCovering(obj) then
                edge ← the graph edge between drone d and this drone;
                edge.weight ← edge.weight + δ;
            end
        end
    end
end
```

References

[1] Sallam Abualhaija, Chetan Arora, Amin Sleimi, and Lionel C. Briand. "Automated Question Answering for Improved Understanding of Compliance Requirements: A Multi-Document Study". In: *2022 IEEE 30th International Requirements Engineering Conference (RE)*. Aug. 2022, pp. 39–50. DOI: 10.1109/RE54965.2022.00011.

[2] Ahmed El Adl. *The emergence of cognitive digital physical twins (CDPT) as the 21st century icons and beacons*. Dec. 2016. URL: https://www.linkedin.com/pulse/emergence-cognitive-digital-physical-twins-cdpt-21st-ahmed/?trk=pulse_spock-articles (visited on 09/14/2023).

[3] Ayyaz Ahmed, Shahid Zulfiqar, Adam Ghandar, Yang Chen, Masatoshi Hanai, and Georgios Theodoropoulos. "Digital Twin Technology for Aquaponics: Towards Optimizing Food Production with Dynamic Data Driven Application Systems". In: *19th Asia Simulation Conference (AsiaSim)*. Oct. 2019, pp. 3–14. ISBN: 978-981-15-1078-6. DOI: 10.1007/978-981-15-1078-6_1.

[4] Fatemeh Akbarian, Emma Fitzgerald, and Maria Kihl. "Synchronization in Digital Twins for Industrial Control Systems". In: *16th Swedish National Computer Networking Workshop (SNCNW 2020)*. May 2020, pp. 1–4. URL: http://arxiv.org/abs/2006.03447.

[5] Okhaide Akhigbe, Daniel Amyot, and Gregory Richards. "A systematic literature mapping of goal and non-goal modelling methods for legal and regulatory compliance". In: *Requirements Engineering* 24.4 (Dec. 2019), pp. 459–481. ISSN: 1432-010X. DOI: 10.1007/s00766-018-0294-1.

[6] Jethro Akroyd, Sebastian Mosbach, Amit Bhave, and Markus Kraft. "Universal Digital Twin - A Dynamic Knowledge Graph". In: *Data-Centric Engineering* 2 (Jan. 2021), e14. ISSN: 2632-6736. DOI: 10.1017/dce.2021.10.

[7] Mohammad Abdullah Al Faruque, Deepan Muthirayan, Shih-Yuan Yu, and Pramod P. Khargonekar. "Cognitive Digital Twin for Manufacturing Systems". In: *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. ISSN: 1558-1101. Feb. 2021, pp. 440–445. DOI: 10.23919/DATE51398.2021.9474166.

[8] Elvin Alberts and Ilias Gerostathopoulos. "Measuring Convergence Inertia: Online Learning in Self-adaptive Systems with Context Shifts". In: *Leveraging Applications of Formal Methods, Verification and Validation. Adaptation and Learning*. Ed. by Tiziana Margaria and Bernhard Steffen. Lecture Notes in Computer Science. Cham: Springer Nature Switzerland, 2022, pp. 231–248. ISBN: 978-3-031-19759-8. DOI: 10.1007/978-3-031-19759-8_15.

[9] Paul Almasan, Miquel Ferriol-Galmés, Jordi Paillisse, José Suárez-Varela, Diego Perino, Diego López, Antonio Agustin Pastor Perales, Paul Harvey, Laurent Ciavaglia, Leon Wong, Vishnu Ram, Shihan Xiao, Xiang Shi, Xiangle Cheng, Albert Cabellos-Aparicio, and Pere Barlet-Ros. "Network Digital Twin: Context, Enabling Technologies, and Opportunities". In: *IEEE Communications Magazine* 60.11 (Nov. 2022), pp. 22–27. ISSN: 1558-1896. DOI: 10.1109/MCOM.001.2200012.

[10] Tomas Ambra and Cathy Macharis. "Agent-Based Digital Twins (ABM-DT) in Synchromodal Transport and Logistics: The Fusion of Virtual and Pysical Spaces". In:

Proceedings of the Winter Simulation Conference. WSC '20. Orlando, FL, USA: IEEE Press, 2020, pp. 159–169. ISBN: 9781728194998.

[11] Daniel Amyot, Sepideh Ghanavati, Jennifer Horkoff, Gunter Mussbacher, Liam Peyton, and Eric Yu. "Evaluating goal models within the goal-oriented requirement language". In: *International Journal of Intelligent Systems* 25.8 (2010), pp. 841–877. ISSN: 1098-111X. DOI: 10.1002/int.20433.

[12] Di An and YangQuan Chen. "Explainable artificial intelligence (XAI) empowered digital twin on soil carbon emission management using proximal sensing". In: *2023 IEEE 3rd International Conference on Digital Twins and Parallel Intelligence (DTPI)*. Nov. 2023, pp. 1–5. DOI: 10.1109/DTPI59677.2023.10365455.

[13] Amal Ahmed Anda. "Modeling Adaptive Socio-Cyber-Physical Systems with Goals and SysML". In: *2018 IEEE 26th International Requirements Engineering Conference (RE)*. Aug. 2018, pp. 442–447. DOI: 10.1109/RE.2018.00059.

[14] ARUP. *Digital twin: towards a meaningful framework*. Tech. rep. ARUP, 2019. URL: www.arup.com/digitaltwinreport.

[15] Behrang Ashtari Talkhestani, Tobias Jung, Benjamin Lindemann, Nada Sahlab, Nasser Jazdi, Wolfgang Schloegl, and Michael Weyrich. "An architecture of an Intelligent Digital Twin in a Cyber-Physical Production System". In: *At-Automatisierungstechnik* 67.9 (2019), pp. 762–782. ISSN: 2196677X. DOI: 10.1515/auto-2019-0039.

[16] Esther van Asselt, Sjoukje Osinga, and Harry Bremmers. "Simulating compliance behaviour for effective inspection strategies using agent based modelling". In: *British Food Journal* 118.4 (Jan. 2016), pp. 809–823. ISSN: 0007-070X. DOI: 10.1108/BFJ-05-2015-0175.

[17] Heiko Aydt, Stephen John Turner, Wentong Cai, and Malcolm Yoke Hean Low. "Research issues in symbiotic simulation". In: *Proceedings of the 2009 Winter Simulation*

Conference (WSC). 2009, pp. 1213–1222. ISBN: 9781424457700. DOI: 10.1109/WSC.2009.5429419.

[18] Heiko Aydt, Stephen John Turner, Wentong Cai, and Malcolm Yoke Hean Low. "Symbiotic simulation systems: An extended definition motivated by symbiosis in biology". In: *2008 22nd Workshop on Principles of Advanced and Distributed Simulation (PADS)*. 2008, pp. 109–116. ISBN: 9780769531595. DOI: 10.1109/PADS.2008.17.

[19] Osman Balci. "Principles and techniques of simulation validation, verification, and testing". In: *Winter Simulation Conference Proceedings* (1995), pp. 147–154. ISSN: 02750708. DOI: 10.1145/224401.224456.

[20] Philip J. Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. "Efficient online reinforcement learning with offline data". In: *Proceedings of the 40th International Conference on Machine Learning*. PMLR, July 2023, pp. 1577–1594. URL: https://proceedings.mlr.press/v202/ball23a.html.

[21] Ahmed Barnawi, Ahmed Awad, Amal Elgammal, Radwa El Shawi, Abdullah Almalaise, and Sherif Sakr. "Runtime self-monitoring approach of business process compliance in cloud environments". In: *Cluster Computing* 18.4 (Dec. 2015), pp. 1503–1526. ISSN: 1573-7543. DOI: 10.1007/s10586-015-0494-0.

[22] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador Garcia, Sergio Gil-Lopez, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. "Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI". In: *Information Fusion* 58 (2020), pp. 82–115. ISSN: 1566-2535. DOI: 10.1016/j.inffus.2019.12.012.

[23] Barbara Rita Barricelli, Elena Casiraghi, and Daniela Fogli. "A survey on digital twin: Definitions, characteristics, applications, and design implications". In: *IEEE Access* 7 (2019), pp. 167653–167671. ISSN: 21693536. DOI: 10.1109/ACCESS.2019.2953499.

[24] Thomas Bauer, Pablo Oliveira Antonino, and Thomas Kuhn. "Towards Architecting Digital Twin-Pervaded Systems". In: *2019 IEEE/ACM 7th International Workshop on Software Engineering for Systems-of-Systems (SESoS) and 13th Workshop on Distributed Software Development, Software Ecosystems and Systems-of-Systems (WDES)*. May 2019, pp. 66–69. ISBN: 9781728134390. DOI: 10.1109/SESoS/WDES.2019.00018.

[25] Emanuele Bellini, Franco Bagnoli, Mauro Caporuscio, Ernesto Damiani, Francesco Flammini, Igor Linkov, Pietro Liò, and Stefano Marrone. "Resilience learning through self adaptation in digital twins of human-cyber-physical systems". In: *2021 IEEE International Conference on Cyber Security and Resilience (CSR)*. July 2021, pp. 168–173. DOI: 10.1109/CSR51186.2021.9527913.

[26] K. Bellman, C. Landauer, N. Dutt, L. Esterle, A. Herkersdorf, A. Jantsch, N. TaheriNejad, P. R. Lewis, M. Platzner, and K. Tammemäe. "Self-aware Cyber-Physical Systems". In: *ACM Transactions on Cyber-Physical Systems* 4.4 (Aug. 2020), pp. 1–26. ISSN: 2378-962X. DOI: 10.1145/3375716.

[27] Nelly Bencomo, Sebastian Götz, and Hui Song. "Models@run.time: a guided tour of the state of the art and research challenges". In: *Software & Systems Modeling* 18.5 (Oct. 2019), pp. 3049–3082. ISSN: 1619-1366. DOI: 10.1007/s10270-018-00712-x.

[28] Alessandra De Benedictis, Francesco Flammini, Nicola Mazzocca, Alessandra Somma, and Francesco Vitale. "Digital Twins for Anomaly Detection in the Industrial Internet of Things: Conceptual Architecture and Proof-of-Concept". In: *IEEE Transactions on Industrial Informatics* (Feb. 2023). Conference Name: IEEE Transactions on Industrial Informatics, pp. 1–11. ISSN: 1941-0050. DOI: 10.1109/TII.2023.3246983.

[29] Pronaya Bhattacharya, Mohammad S. Obaidat, Sakshi Sanghavi, Vatsal Sakariya, Sudeep Tanwar, and Kuei-Fang Hsiao. "Internet-of-explainable-digital-twins: A case study of versatile corn production ecosystem". In: *2022 International Conference*

on Communications, Computing, Cybersecurity, and Informatics (CCCI). Oct. 2022, pp. 1–5. DOI: 10.1109/CCCI55352.2022.9926502.

[30] Ani Bicaku, Markus Tauber, and Jerker Delsing. "Security standard compliance and continuous verification for Industrial Internet of Things". In: *International Journal of Distributed Sensor Networks* 16.6 (June 2020), pp. 1–19. ISSN: 1550-1329. DOI: 10.1177/1550147720922731.

[31] Ani Bicaku, Mario Zsilak, Peter Theiler, Markus Tauber, and Jerker Delsing. "Security Standard Compliance Verification in System of Systems". In: *IEEE Systems Journal* 16.2 (2022), pp. 2195–2205. DOI: 10.1109/JSYST.2021.3064196.

[32] A. Biere, A. Cimatti, E.M. Clarke, M. Fujita, and Y. Zhu. "Symbolic Model Checking using SAT procedures instead of BDDs". In: *Proceedings 1999 Design Automation Conference (Cat. No. 99CH36361)*. New York, NY, USA: ACM, June 1999, pp. 317–320. DOI: 10.1109/DAC.1999.781333.

[33] Aliaksandr Birukou, Vincenzo D'Andrea, Frank Leymann, Jacek Serafinski, Patricia Silveira, Steve Strauch, and Marek Tluczek. "An Integrated Solution for Runtime Compliance Governance in SOA". In: *8th International Conference on Service-Oriented Computing (ICSOC)*. Dec. 2010, pp. 122–136. ISBN: 978-3-642-17358-5. DOI: 10.1007/978-3-642-17358-5_9.

[34] Erik Blasch. "Dddas Advantages from High-Dimensional Simulation". In: *2018 Winter Simulation Conference (WSC)*. 2018, pp. 1418–1429. DOI: 10.1109/WSC.2018.8632336.

[35] Avrim Blum. "On-line algorithms in machine learning". In: *Online Algorithms: The State of the Art*. Ed. by Amos Fiat and Gerhard J. Woeginger. Berlin, Heidelberg: Springer, 1998, pp. 306–325. ISBN: 978-3-540-68311-7. DOI: 10.1007/BFb0029575.

[36] Nicola Bombieri, Franco Fummi, Graziano Pravadelli, and Joao Marques-Silva. "Towards Equivalence Checking Between TLM and RTL Models". In: *2007 5th IEEE/ACM International Conference on Formal Methods and Models for Codesign (MEMOCODE 2007)*. USA: IEEE Computer Society, May 2007, pp. 113–122.

[37] Stefan Boschert and Roland Rosen. "Digital Twin—The Simulation Aspect". In: *Mechatronic Futures: Challenges and Solutions for Mechatronic Systems and their Designers*. Cham: Springer International Publishing, 2016, pp. 59–74. ISBN: 978-3-319-32156-1. DOI: 10.1007/978-3-319-32156-1_5.

[38] Travis D. Breaux, Matthew W. Vail, and Annie I. Anton. "Towards Regulatory Compliance: Extracting Rights and Obligations to Align Requirements with Regulations". In: *14th IEEE International Requirements Engineering Conference (RE'06)*. Sept. 2006, pp. 49–58. DOI: 10.1109/RE.2006.68.

[39] Tomas Bures, Danny Weyns, Bradley Schmer, Eduardo Tovar, Eric Boden, Thomas Gabor, Ilias Gerostathopoulos, Pragya Gupta, Eunsuk Kang, Alessia Knauss, Pankesh Patel, Awais Rashid, Ivan Ruchkin, Roykrong Sukkerd, and Christos Tsigkanos. "Software Engineering for Smart Cyber-Physical Systems: Challenges and Promising Solutions". In: *ACM SIGSOFT Software Engineering Notes* 42.2 (June 2017), pp. 19–24. ISSN: 0163-5948. DOI: 10.1145/3089649.3089656.

[40] Kun Cao, Shiyan Hu, Yang Shi, Armando Walter Colombo, Stamatis Karnouskos, and Xin Li. "A Survey on Edge and Edge-Cloud Computing Assisted Cyber-Physical Systems". In: *IEEE Transactions on Industrial Informatics* 17.11 (2021), pp. 7806–7819. DOI: 10.1109/TII.2021.3073066.

[41] Yiyun Cao, Christine Currie, Bhakti Stephan Onggo, and Michael Higgins. "Simulation Optimization for a Digital Twin Using a Multi-Fidelity Framework". In: *Proceedings of the Winter Simulation Conference*. WSC '21. IEEE Press, 2021.

[42] Valeria Cardellini, Francesco Lo Presti, Matteo Nardelli, and Gabriele Russo Russo. "Decentralized self-adaptation for elastic Data Stream Processing". In: *Future Generation Computer Systems* 87 (2018), pp. 171–185. ISSN: 0167-739X.

[43] John S. Carson. "Model Verification and Validation". In: *Proceedings of the Winter Simulation Conference*. Vol. 1. San Diego, CA, USA: IEEE Press, 2002, 52–58 vol.1.

[44] Roberto Casadei, Andrea Placuzzi, Mirko Viroli, and Danny Weyns. "Augmented Collective Digital Twins for Self-Organising Cyber-Physical Systems". In: *2021 IEEE International Conference on Autonomic Computing and Self-Organizing Systems Companion (ACSOS-C)*. Sept. 2021, pp. 160–165. DOI: 10.1109/ACSOS-C52956.2021.00051.

[45] Gregory Chance, Dhaminda B. Abeywickrama, Beckett LeClair, Owen Kerr, and Kerstin Eder. *Assessing trustworthiness of autonomous systems*. May 2023. DOI: 10.48550/arXiv.2305.03411.

[46] Shuyi Chen, Masatoshi Hanai, Zhengchang Hua, Nikos Tziritas, and Georgios Theodoropoulos. "Efficient Direct Agent Interaction in Optimistic Distributed Multi-Agent-System Simulations". In: *Proceedings of the 2020 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*. SIGSIM-PADS '20. Miami, FL, Spain: ACM, 2020, pp. 123–128. ISBN: 9781450375924.

[47] Tao Chen. "All versus one: An empirical comparison on retrained and incremental machine learning for modeling performance of adaptable software". In: *2019 IEEE/ACM 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. May 2019, pp. 157–168. DOI: 10.1109/SEAMS.2019.00029.

[48] Tao Chen, Rami Bahsoon, and Xin Yao. "A Survey and Taxonomy of Self-Aware and Self-Adaptive Cloud Autoscaling Systems". In: *ACM Computing Surveys* 51.3 (July 2018), pp. 1–40. ISSN: 0360-0300. DOI: 10.1145/3190507.

[49] Tao Chen, Funmilade Faniyi, Rami Bahsoon, Peter R. Lewis, Xin Yao, Leandro L. Minku, and Lukas Esterle. *The Handbook of Engineering Self-Aware and Self-Expressive Systems*. 2015. URL: http://arxiv.org/abs/1409.1793v3.

[50] Zhiyuan Chen and Bing Liu. *Lifelong machine learning*. Second edition. Synthesis lectures on artificial intelligence and machine learning 38. San Rafael, California: Morgan & Claypool Publishers, 2018. ISBN: 978-1-68173-303-6.

[51] Betty H. C. Cheng, Rogério de Lemos, Holger Giese, Paola Inverardi, Jeff Magee, Jesper Andersson, Basil Becker, Nelly Bencomo, Yuriy Brun, Bojan Cukic, Giovanna Di Marzo Serugendo, Schahram Dustdar, Anthony Finkelstein, Cristina Gacek, Kurt Geihs, Vincenzo Grassi, Gabor Karsai, Holger M. Kienle, Jeff Kramer, Marin Litoiu, Sam Malek, Raffaela Mirandola, Hausi A. Müller, Sooyong Park, Mary Shaw, Matthias Tichy, Massimo Tivoli, Danny Weyns, and Jon Whittle. "Software Engineering for Self-Adaptive Systems: A Research Roadmap". In: *Software Engineering for Self-Adaptive Systems*. Ed. by Betty H. C. Cheng, Rogério de Lemos, Holger Giese, Paola Inverardi, and Jeff Magee. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 1–26. ISBN: 978-3-642-02161-9. DOI: 10.1007/978-3-642-02161-9_1.

[52] Kushan Choksi, Abdul Basit Mirza, Austin Zhou, Deepi Singh, Masayuki Hijikata, and Fang Luo. "Self-Evolving Digital Twin-based Online Health Monitoring of Multi-Phase Boost Converters". In: *IEEE Transactions on Power Electronics* (2023), pp. 1–16. ISSN: 1941-0107. DOI: 10.1109/TPEL.2023.3311710.

[53] Berkeley Churchill, Oded Padon, Rahul Sharma, and Alex Aiken. "Semantic Program Alignment for Equivalence Checking". In: *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*. PLDI 2019. Phoenix, AZ, USA: ACM, 2019, pp. 1027–1040. ISBN: 9781450367127.

[54] Edmund M. Clarke, William Klieber, Miloš Nováček, and Paolo Zuliani. "Model Checking and the State Explosion Problem". In: *Tools for Practical Software Ver-*

ification: LASER, International Summer School 2011, Elba Island, Italy, Revised Tutorial Lectures. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 1–30. ISBN: 978-3-642-35746-6.

[55] Thomas Clemen, Nima Ahmady-Moghaddam, Ulfia A Lenfers, Florian Ocker, Daniel Osterholz, Jonathan Ströbele, and Daniel Glake. "Multi-Agent Systems and Digital Twins for Smarter Cities". In: *Proceedings of the 2021 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*. SIGSIM-PADS '21. New York, NY, USA: ACM, 2021, pp. 45–55. ISBN: 9781450382960.

[56] C Cronrath, A R Aderiani, and B Lennartson. "Enhancing Digital Twins through Reinforcement Learning". In: *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*. Aug. 2019, pp. 293–298. DOI: 10.1109/COASE.2019.8842888.

[57] Rosario Davide D'Amico, John Ahmet Erkoyuncu, Sri Addepalli, and Steve Penver. "Cognitive digital twin: An approach to improve the maintenance management". In: *CIRP Journal of Manufacturing Science and Technology* 38 (Aug. 2022), pp. 613–630. ISSN: 1755-5817. DOI: 10.1016/j.cirpj.2022.06.004.

[58] Hung V. Dang, Mallik Tatipamula, and Huan X. Nguyen. "Cloud-Based Digital Twinning for Structural Health Monitoring Using Deep Learning". In: *IEEE Transactions on Industrial Informatics* 18.6 (June 2022), pp. 3820–3830. ISSN: 1941-0050. DOI: 10.1109/TII.2021.3115119.

[59] Anne Dardenne, Axel van Lamsweerde, and Stephen Fickas. "Goal-directed requirements acquisition". In: *Science of Computer Programming* 20.1 (Apr. 1993), pp. 3–50. ISSN: 0167-6423. DOI: 10.1016/0167-6423(93)90021-G.

[60] Frederica Darema. "Dynamic Data Driven Applications Systems: A New Paradigm for Application Simulations and Measurements". In: *Computational Science - ICCS 2004*. Ed. by Marian Bubak, Geert Dick van Albada, Peter M. A. Sloot, and Jack

Dongarra. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 662–669. ISBN: 978-3-540-24688-6. DOI: 10.1007/978-3-540-24688-6_86.

[61] Nicoletta De Francesco, Giuseppe Lettieri, Antonella Santone, and Gigliola Vaglini. "Heuristic Search for Equivalence Checking". In: *Software & Systems Modeling* 15.2 (2016), pp. 513–530. ISSN: 1619-1374.

[62] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. "A Continual Learning Survey: Defying Forgetting in Classification Tasks". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.7 (July 2022). Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 3366–3385. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2021.3057446.

[63] Mohammad Dehghanimohammadabadi, Sahil Belsare, and Renee Thiesing. "Simulation-Optimization of Digital Twin". In: *2021 Winter Simulation Conference (WSC)*. Dec. 2021, pp. 1–10. DOI: 10.1109/WSC52266.2021.9715412.

[64] Juan Deng, Qingbi Zheng, Guangyi Liu, Jielin Bai, Kaicong Tian, Changhao Sun, Yujie Yan, and Yitong Liu. "A Digital Twin Approach for Self-optimization of Mobile Networks". In: *2021 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*. Nanjing, China: IEEE, Mar. 2021, pp. 1–6. ISBN: 978-1-72819-507-0. DOI: 10.1109/WCNCW49093.2021.9420037.

[65] Ada Diaconescu, Kirstie L. Bellman, Lukas Esterle, Holger Giese, Sebastian Götz, Peter Lewis, and Andrea Zisman. "Architectures for Collective Self-aware Computing Systems". In: *Self-Aware Computing Systems*. Cham, Switzerland: Springer International Publishing, 2017, pp. 191–235. ISBN: 9783319474748.

[66] Georgios Diamantopoulos, Nikos Tziritas, Rami Bahsoon, and Georgios Theodoropoulos. "Digital Twins for Dynamic Management of Blockchain Systems". In: *2022*

Winter Simulation Conference (WSC). To Appear. Apr. 2022. URL: http://arxiv.org/abs/2204.12477.

[67] Marietheres Dietz, Manfred Vielberth, and Günther Pernul. "Integrating Digital Twin Security Simulations in the Security Operations Center". In: *Proceedings of the 15th International Conference on Availability, Reliability and Security*. ARES '20. New York, NY, USA: Association for Computing Machinery, 2020. ISBN: 9781450388337. DOI: 10.1145/3407023.3407039.

[68] Matthias Eckhart and Andreas Ekelhart. "A Specification-based State Replication Approach for Digital Twins". In: *Proceedings of the 2018 Workshop on Cyber-Physical Systems Security and PrivaCy*. CPS-SPC '18. New York, New York, USA: ACM Press, 2018, pp. 36–47. ISBN: 9781450359924.

[69] A. Elhabbash, R. Bahsoon, P. Tino, P. R. Lewis, and Y. Elkhatib. "Attaining Meta-self-awareness through Assessment of Quality-of-Knowledge". In: *2021 IEEE International Conference on Web Services (ICWS)*. Los Alamitos, CA, USA: IEEE Computer Society, Sept. 2021, pp. 712–723.

[70] Abdessalam Elhabbash, Rami Bahsoon, and Peter Tino. "Interaction-Awareness for Self-Adaptive Volunteer Computing". In: *2016 IEEE 10th International Conference on Self-Adaptive and Self-Organizing Systems (SASO)*. Augsburg, Germany: IEEE Press, 2016, pp. 148–149.

[71] Abdessalam Elhabbash, Maria Salama, Rami Bahsoon, and Peter Tino. "Self-awareness in Software Engineering: A Systematic Literature Review". In: *ACM Transactions on Autonomous and Adaptive Systems* 14.2 (Oct. 2019), pp. 1–42. ISSN: 15564665. DOI: 10.1145/3347269.

[72] Naeem Esfahani and Sam Malek. "Uncertainty in Self-Adaptive Software Systems". In: *Software Engineering for Self-Adaptive Systems II: International Seminar, Dagstuhl Castle, Germany, October 24-29, 2010 Revised Selected and Invited Papers*. Vol. 7475

LNCS. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 214–238. ISBN: 978-3-642-35813-5.

[73] Lukas Esterle and John N A Brown. "I Think Therefore You Are: Models for Interaction in Collectives of Self-Aware Cyber-Physical Systems". In: *ACM Trans. Cyber-Phys. Syst.* 4.4 (2020). ISSN: 2378-962X. DOI: 10.1145/3375403.

[74] Lukas Esterle and Peter R Lewis. "Online Multi-Object k-Coverage with Mobile Smart Cameras". In: *Proceedings of the 11th International Conference on Distributed Smart Cameras*. ICDSC 2017. New York, NY, USA: ACM, 2017, pp. 107–112. ISBN: 9781450354875.

[75] Lukas Esterle, Peter R. Lewis, Xin Yao, and Bernhard Rinner. "Socio-economic vision graph generation and handover in distributed smart camera networks". In: *ACM Transactions on Sensor Networks* 10.2 (2014), pp. 1–24. ISSN: 15504859. DOI: 10.1145/2530001.

[76] Felix Feit, Andreas Metzger, and Klaus Pohl. "Explaining online reinforcement learning decisions of self-adaptive systems". In: *2022 IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS)*. Sept. 2022, pp. 51–60. DOI: 10.1109/ACSOS55765.2022.00023.

[77] Hao Feng, Cláudio Gomes, Santiago Gil, Peter H. Mikkelsen, Daniella Tola, Peter Gorm Larsen, and Michael Sandberg. "Integration Of The Mape-K Loop In Digital Twins". In: *2022 Annual Modeling and Simulation Conference (ANNSIM)*. July 2022, pp. 102–113. DOI: 10.23919/ANNSIM55834.2022.9859489.

[78] Francesco Flammini. "Digital twins as run-time predictive models for the resilience of cyber-physical systems: a conceptual framework". In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 379.2207 (Oct. 2021), p. 11. DOI: 10.1098/rsta.2020.0369.

[79] Stan Franklin, Tamas Madl, Sidney D'Mello, and Javier Snaider. "LIDA: A systems-level architecture for cognition, emotion, and learning". In: *IEEE Transactions on Autonomous Mental Development* 6.1 (Mar. 2014), pp. 19–41. ISSN: 1943-0612. DOI: 10.1109/TAMD.2013.2277589.

[80] Mirgita Frasheri, Henrik Ejersbo, Casper Thule, Cláudio Gomes, Jakob Levisen Kvistgaard, Peter Gorm Larsen, and Lukas Esterle. "Addressing time discrepancy between digital and physical twins". In: *Robotics and Autonomous Systems* 161 (Mar. 2023), p. 104347. ISSN: 0921-8890. DOI: 10.1016/j.robot.2022.104347.

[81] Richard Fujimoto, Dell Lunceford, Ernest Page, and Adelinde M. Uhrmacher. *Grand Challenges for Modeling and Simulation Dagstuhl Report*. Tech. rep. 2002.

[82] Richard M. Fujimoto. *Parallel and distribution simulation systems*. New York: Wiley, 2000. ISBN: 978-0-471-18383-9.

[83] Thomas Gabor, Lenz Belzner, Marie Kiermeier, Michael Till Beck, and Alexander Neitz. "A Simulation-Based Architecture for Smart Cyber-Physical Systems". In: *2016 IEEE International Conference on Autonomic Computing (ICAC)*. IEEE, July 2016, pp. 374–379. ISBN: 978-1-5090-1654-9. DOI: 10.1109/ICAC.2016.29.

[84] Chuanchao Gao, Heejong Park, and Arvind Easwaran. "An Anomaly Detection Framework for Digital Twin Driven Cyber-Physical Systems". In: *Proceedings of the ACM/IEEE 12th International Conference on Cyber-Physical Systems*. ICCPS '21. New York, NY, USA: ACM, 2021, pp. 44–54. ISBN: 9781450383530.

[85] Chuanchao Gao, Heejong Park, and Arvind Easwaran. "An Anomaly Detection Framework for Digital Twin Driven Cyber-Physical Systems". In: *Proceedings of the ACM/IEEE 12th International Conference on Cyber-Physical Systems*. ICCPS '21. Nashville, Tennessee: ACM, 2021, pp. 44–54. ISBN: 9781450383530. DOI: 10.1145/3450267.3450533.

[86] Jesús García-Galán, Liliana Pasquale, George Grispos, and Bashar Nuseibeh. "Towards Adaptive Compliance". In: *2016 IEEE/ACM 11th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. 2016, pp. 108–114. DOI: 10.1109/SEAMS.2016.020.

[87] Sepideh Ghanavati, Daniel Amyot, and André Rifaut. "Legal goal-oriented requirement language (legal GRL) for modeling regulations". In: *Proceedings of the 6th International Workshop on Modeling in Software Engineering*. MiSE 2014. June 2014, pp. 1–6. ISBN: 978-1-4503-2849-4. DOI: 10.1145/2593770.2593780.

[88] Sepideh Ghanavati and Joris Hulstijn. "Impact of Legal Interpretation in Business Process Compliance". In: *2015 IEEE/ACM 1st International Workshop on TEchnical and LEgal aspects of data pRivacy and SEcurity*. May 2015, pp. 26–31. DOI: 10.1109/TELERISE.2015.13.

[89] Adam Ghandar, Ayyaz Ahmed, Shahid Zulfiqar, Zhengchang Hua, Masatoshi Hanai, and Georgios Theodoropoulos. "A Decision Support System for Urban Agriculture Using Digital Twin: A Case Study With Aquaponics". In: *IEEE Access* 9 (2021), pp. 35691–35708.

[90] Omid Gheibi and Danny Weyns. "Dealing with drift of adaptation spaces in learning-based self-adaptive systems using lifelong self-adaptation". In: *ACM Transactions on Autonomous and Adaptive Systems* 19.1 (Feb. 2024), 5:1–5:57. ISSN: 1556-4665. DOI: 10.1145/3636428. (Visited on 04/19/2024).

[91] Omid Gheibi and Danny Weyns. "Lifelong self-adaptation: self-adaptation meets lifelong machine learning". In: *Proceedings of the 17th Symposium on Software Engineering for Adaptive and Self-Managing Systems*. SEAMS '22. New York, NY, USA: Association for Computing Machinery, Aug. 2022, pp. 1–12. ISBN: 978-1-4503-9305-8. DOI: 10.1145/3524844.3528052.

[92] C. Lee Giles, Christian W. Omlin, and Karvel K. Thornber. "Equivalence in Knowledge Representation: Automata, Recurrent Neural Networks, and Dynamical Fuzzy Systems". In: *Proceedings of the IEEE* 87.9 (1999), pp. 1623–1640.

[93] Edward Glaessgen and David Stargel. "The Digital Twin Paradigm for Future NASA and U.S. Air Force Vehicles". In: *53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*. Apr. 2012, pp. 1–14. ISBN: 978-1-60086-937-2. DOI: 10.2514/6.2012-1818.

[94] Michael Grieves and John Vickers. "Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems". In: *Transdisciplinary Perspectives on Complex Systems: New Findings and Approaches*. Ed. by Franz-Josef Kahlen, Shannon Flumerfelt, and Anabela Alves. Cham: Springer International Publishing, 2017, pp. 85–113. ISBN: 978-3-319-38756-7. DOI: 10.1007/978-3-319-38756-7_4.

[95] Sven Gronauer and Klaus Diepold. "Multi-agent deep reinforcement learning: a survey". In: *Artificial Intelligence Review* 55.2 (Feb. 2022), pp. 895–943. ISSN: 1573-7462.

[96] Britton Hammit, Rachel James, and Mohamed Ahmed. "A Case for Online Traffic Simulation: Systematic Procedure to Calibrate Car-Following Models Using Vehicle Data". In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. Maui, HI, USA: IEEE Press, Nov. 2018, pp. 3785–3790.

[97] Hossein Hashemi, Khaled F. Abdelghany, and Ahmed F. Abdelghany. "A Multi-Agent Learning Approach for Online Calibration and Consistency Checking of Real-Time Traffic Network Management Systems". In: *Transportmetrica B: Transport Dynamics* 5.3 (July 2017), pp. 364–384. ISSN: 2168-0566.

[98] Mustafa Hashmi, Guido Governatori, Ho-Pun Lam, and Moe Thandar Wynn. "Are we done with business process compliance: state of the art and challenges ahead". In: *Knowledge and Information Systems* 57.1 (Oct. 2018), pp. 79–133. ISSN: 0219-3116. DOI: 10.1007/s10115-017-1142-1.

[99] Dwayne Henclewood, Wonho Suh, Michael Rodgers, Michael Hunter, and Richard Fujimoto. "A Case for Real-Time Calibration of Data-Driven Microscopic Traffic Simulation Tools". In: *Proceedings of the 2012 Winter Simulation Conference (WSC)*. Berlin, Germany: IEEE Press, Dec. 2012, pp. 1–12.

[100] Lukas-Valentin Herm, Kai Heinrich, Jonas Wanner, and Christian Janiesch. "Stop ordering machine learning algorithms by their explainability! A user-centered investigation of performance and explainability". In: *International Journal of Information Management* (June 2022), p. 102538. ISSN: 0268-4012. DOI: 10.1016/j.ijinfomgt.2022.102538.

[101] Henry Hoffmann, Axel Jantsch, and Nikil D. Dutt. "Embodied Self-Aware Computing Systems". In: *Proceedings of the IEEE* 108.7 (July 2020). Conference Name: Proceedings of the IEEE, pp. 1027–1046. ISSN: 1558-2256. DOI: 10.1109/JPROC.2020.2977054.

[102] Steven C. H. Hoi, Doyen Sahoo, Jing Lu, and Peilin Zhao. "Online learning: A comprehensive survey". In: *Neurocomputing* 459 (Oct. 2021), pp. 249–289. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2021.04.112.

[103] Karl Hribernik, Giacomo Cabri, Federica Mandreoli, and Gregoris Mentzas. "Autonomous, context-aware, adaptive Digital Twins—State of the art and roadmap". In: *Computers in Industry* 133 (Dec. 2021), p. 103508. ISSN: 01663615. DOI: 10.1016/j.compind.2021.103508.

[104] Guoqiang Hu, Wee Peng Tay, and Yonggang Wen. "Cloud robotics: architecture, challenges and applications". In: *IEEE Network* 26.3 (2012), pp. 21–28. DOI: 10.1109/MNET.2012.6201212.

[105] Xiaolin Hu and Peisheng Wu. "A Data Assimilation Framework for Discrete Event Simulations". In: *ACM Trans. Model. Comput. Simul.* 29.3 (June 2019). ISSN: 1049-3301. DOI: 10.1145/3301502.

[106] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. *A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions*. Nov. 2023. DOI: 10.48550/arXiv.2311.05232. (Visited on 04/16/2024).

[107] Markus C. Huebscher and Julie A. McCann. "A Survey of Autonomic Computing—Degrees, Models, and Applications". In: *ACM Computing Surveys* 40.3 (Aug. 2008), pp. 1–28. ISSN: 03600300. DOI: 10.1145/1380584.1380585.

[108] IBM. *An architectural blueprint for autonomic computing*. Tech. rep. 2005, p. 34.

[109] IBM. *Autonomic computing: IBM's perspective on the state of information technology*. Tech. rep. 2001.

[110] IEEE Std 7001-2021. *IEEE Standard for Transparency of Autonomous Systems*. Standard. Mar. 2022. URL: https://doi.org/10.1109/IEEESTD.2022.9726144.

[111] IEEE Std 7010-2020. *IEEE Recommended Practice for Assessing the Impact of Autonomous and Intelligent Systems on Human Well-Being*. Standard. May 2020. URL: https://doi.org/10.1109/IEEESTD.2020.9084219.

[112] Silvia Ingolfo, Alberto Siena, John Mylopoulos, Angelo Susi, and Anna Perini. "Arguing regulatory compliance of software requirements". In: *Data & Knowledge Engineering* 87 (Sept. 2013), pp. 279–296. ISSN: 0169-023X. DOI: 10.1016/j.datak.2012.12.004.

[113] ISO 3691-4:2020. *Industrial trucks — Safety requirements and verification — Part 4: Driverless industrial trucks and their systems*. Standard. Geneva, CH: International Organization for Standardization, Feb. 2020. URL: https://www.iso.org/standard/83545.html.

[114] Pooyan Jamshidi, Amir M. Sharifloo, Claus Pahl, Andreas Metzger, and Giovani Estrada. "Self-Learning Cloud Controllers: Fuzzy Q-Learning for Knowledge Evo-

lution". In: *Proceedings - 2015 International Conference on Cloud and Autonomic Computing, ICCAC 2015* (2015), pp. 208–211. DOI: 10.1109/ICCAC.2015.35.

[115] Pengyi Jia, Xianbin Wang, and Xuemin Shen. "Digital-Twin-Enabled Intelligent Distributed Clock Synchronization in Industrial IoT Systems". In: *IEEE Internet of Things Journal* 8.6 (2021), pp. 4548–4559.

[116] Lu Jinzhi, Yang Zhaorui, Zheng Xiaochen, Wang Jian, and Kiritsis Dimitris. "Exploring the concept of cognitive digital twin from model-based systems engineering perspective". In: *The International Journal of Advanced Manufacturing Technology* 121.9 (Aug. 2022), pp. 5835–5854. ISSN: 1433-3015. DOI: 10.1007/s00170-022-09610-5.

[117] David Jones, Chris Snider, Aydin Nassehi, Jason Yon, and Ben Hicks. "Characterising the Digital Twin: A Systematic Literature Review". In: *CIRP Journal of Manufacturing Science and Technology* 29 (2020), pp. 36–52. ISSN: 17555817.

[118] Deepti Kalasapura, Jinyang Li, Shengzhong Liu, Yizhuo Chen, Ruijie Wang, Tarek Abdelzaher, Matthew Caesar, Joydeep Bhattacharyya, Jae Kim, Guijun Wang, Greg Kimberly, Josh Eckhardt, and Denis Osipychev. "TwinSync: A Digital Twin Synchronization Protocol for Bandwidth-Limited IoT Applications". In: *2023 32nd International Conference on Computer Communications and Networks (ICCCN)*. ISSN: 2637-9430. July 2023, pp. 1–1. DOI: 10.1109/ICCCN58024.2023.10230154.

[119] Michael G. Kapteyn, David J. Knezevic, and Karen Willcox. "Toward predictive Digital Twins via component-based reduced-order models and interpretable machine learning". In: *AIAA Scitech 2020 Forum*. Reston, Virginia: American Institute of Aeronautics and Astronautics, Jan. 2020, pp. 1–19. ISBN: 978-1-62410-595-1. DOI: 10.2514/6.2020-0418.

[120] Michael G. Kapteyn, Jacob V. R. Pretorius, and Karen E. Willcox. "A Probabilistic Graphical Model Foundation for Enabling Predictive Digital Twins at Scale". In: *Nature Computational Science* 1.5 (May 2021), pp. 337–347. ISSN: 2662-8457.

[121] A Karakra, F Fontanili, E Lamine, and J 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)*. May 2019, pp. 1–4. DOI: 10.1109/BHI.2019.8834534.

[122] A Karakra, F Fontanili, E Lamine, J Lamothe, and A Taweel. "Pervasive Computing Integrated Discrete Event Simulation for a Hospital Digital Twin". In: *2018 IEEE/ACS 15th International Conference on Computer Systems and Applications (AICCSA)*. Oct. 2018, pp. 1–6. DOI: 10.1109/AICCSA.2018.8612796.

[123] S. Kate Devitt. "Trustworthiness of autonomous systems". In: *Foundations of Trusted Autonomy*. Ed. by Hussein A. Abbass, Jason Scholz, and Darryn J. Reid. Cham: Springer International Publishing, 2018, pp. 161–184. ISBN: 978-3-319-64816-3. DOI: 10.1007/978-3-319-64816-3_9.

[124] Rick Kazman, Mark Klein, and Paul Clements. *ATAM: Method for Architecture Evaluation*. Tech. rep. CMU/SEI-2000-TR-004. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2000. URL: http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=5177.

[125] Catriona Kennedy and Georgios Theodoropoulos. "Intelligent Management of Data Driven Simulations to Support Model Building in the Social Sciences". In: *Computational Science – ICCS 2006*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 562–569. DOI: 10.1007/11758532_74.

[126] Catriona Kennedy, Georgios Theodoropoulos, Volker Sorge, Edward Ferrari, Peter Lee, and Chris Skelcher. "AIMSS: An Architecture for Data Driven Simulations in the Social Sciences". In: *Computational Science – ICCS 2007*. Lecture Notes in Computer

Science. Berlin, Heidelberg: Springer, 2007, pp. 1098–1105. ISBN: 978-3-540-72584-8. DOI: 10.1007/978-3-540-72584-8_144.

[127] Catriona Kennedy, Georgios Theodoropoulos, Volker Sorge, Edward Ferrari, Peter Lee, and Chris Skelcher. "Data Driven Simulation to Support Model Building in the Social Sciences". In: *Journal of Algorithms & Computational Technology* 5.4 (2011), pp. 561–581. DOI: 10.1260/1748-3018.5.4.561.

[128] J.O. Kephart and D.M. Chess. "The vision of autonomic computing". In: *Computer* 36.1 (Jan. 2003), pp. 41–50. ISSN: 1558-0814. DOI: 10.1109/MC.2003.1160055.

[129] Khimya Khetarpal, Matthew Riemer, Irina Rish, and Doina Precup. "Towards Continual Reinforcement Learning: A Review and Perspectives". In: *Journal of Artificial Intelligence Research* 75 (Dec. 2022), pp. 1401–1476. ISSN: 1076-9757. DOI: 10.1613/jair.1.13673.

[130] R. Kim and P.M. Torrens. "Building verisimilitude in VR with high-fidelity local action models: a demonstration supporting road-crossing experiments". In: *38th ACM SIGSIM Conference on Principles of Advanced Discrete Simulation (PADS '24)*. Atlanta GA USA, June 2024, pp. 1–17.

[131] David Knuplesch, Manfred Reichert, and Akhil Kumar. "A framework for visually monitoring business process compliance". In: *Information Systems* 64 (Mar. 2017), pp. 381–409. ISSN: 0306-4379. DOI: 10.1016/j.is.2016.10.006.

[132] Kazuma Kobayashi and Syed Bahauddin Alam. "Explainable, interpretable, and trustworthy AI for an intelligent digital twin: A case study on remaining useful life". In: *Engineering Applications of Artificial Intelligence* 129 (Mar. 2024), p. 20. ISSN: 0952-1976. DOI: 10.1016/j.engappai.2023.107620.

[133] William Koch, Renato Mancuso, Richard West, and Azer Bestavros. "Reinforcement Learning for UAV Attitude Control". In: *ACM Transactions on Cyber-Physical Systems* 3.2 (Feb. 2019), pp. 1–21. ISSN: 2378962X. DOI: 10.1145/3301273.

[134] B Korth, C Schwede, and M Zajac. "Simulation-Ready Digital Twin for Realtime Management of Logistics Systems". In: *2018 IEEE International Conference on Big Data (Big Data)*. Seattle, WA, USA: IEEE Press, Dec. 2018, pp. 4194–4201.

[135] Samuel Kounev, Peter Lewis, Kirstie L Bellman, Nelly Bencomo, Javier Camara, Ada Diaconescu, Lukas Esterle, Kurt Geihs, Holger Giese, Sebastian Götz, Paola Inverardi, Jeffrey O Kephart, and Andrea Zisman. "The Notion of Self-aware Computing". In: *Self-Aware Computing Systems*. Ed. by Samuel Kounev, Jeffrey O Kephart, Aleksandar Milenkoski, and Xiaoyun Zhu. Cham: Springer International Publishing, 2017, pp. 3–16. ISBN: 978-3-319-47474-8. DOI: 10.1007/978-3-319-47474-8_1.

[136] Shitij Kumar, Celal Savur, and Ferat Sahin. "Survey of Human–Robot Collaboration in Industrial Settings: Awareness, Intelligence, and Compliance". In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 51.1 (2021), pp. 280–297. ISSN: 2168-2232. DOI: 10.1109/TSMC.2020.3041231.

[137] A. van Lamsweerde. "Goal-oriented requirements engineering: A guided tour". In: *Proceedings Fifth IEEE International Symposium on Requirements Engineering*. Aug. 2001, pp. 249–262. DOI: 10.1109/ISRE.2001.948567. (Visited on 03/29/2024).

[138] Michael Lees, Brian Logan, Rob Minson, Ton Oguara, and Georgios Theodoropoulos. "Modelling Environments for Distributed Simulation". In: *Environments for Multi-Agent Systems*. Ed. by Danny Weyns, H. Van Dyke Parunak, and Fabien Michel. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 150–167. ISBN: 978-3-540-32259-7.

[139] L Lei, G Shen, L Zhang, and Z Li. "Toward Intelligent Cooperation of UAV Swarm: When Machine Learning Meets Digital Twin". In: *IEEE Network* (2020), pp. 1–7. ISSN: 1558-156X. DOI: 10.1109/MNET.011.2000388.

[140] Paulo Leitão, Stamatis Karnouskos, Luis Ribeiro, Jay Lee, Thomas Strasser, and Armando W. Colombo. "Smart Agents in Industrial Cyber-Physical Systems". In: *Proceedings of the IEEE* 104.5 (2016), pp. 1086–1101. ISSN: 15582256. DOI: 10.1109/JPROC.2016.2521931.

[141] Rogério de Lemos, Holger Giese, Hausi A Müller, Mary Shaw, Jesper Andersson, Marin Litoiu, Bradley Schmerl, Gabriel Tamura, Norha M Villegas, Thomas Vogel, Danny Weyns, Luciano Baresi, Basil Becker, Nelly Bencomo, Yuriy Brun, Bojan Cukic, Ron Desmarais, Schahram Dustdar, Gregor Engels, Kurt Geihs, Karl M Göschka, Alessandra Gorla, Vincenzo Grassi, Paola Inverardi, Gabor Karsai, Jeff Kramer, Antónia Lopes, Jeff Magee, Sam Malek, Serge Mankovskii, Raffaela Mirandola, John Mylopoulos, Oscar Nierstrasz, Mauro Pezzè, Christian Prehofer, Wilhelm Schäfer, Rick Schlichting, Dennis B Smith, João Pedro Sousa, Ladan Tahvildari, Kenny Wong, and Jochen Wuttke. "Software Engineering for Self-Adaptive Systems: A Second Research Roadmap". In: *Software Engineering for Self-Adaptive Systems II: International Seminar, Dagstuhl Castle, Germany, October 24-29, 2010 Revised Selected and Invited Papers*. Springer Berlin Heidelberg, 2013, pp. 1–32. ISBN: 978-3-642-35813-5. DOI: 10.1007/978-3-642-35813-5_1.

[142] Peter R. Lewis, Arjun Chandra, Funmilade Faniyi, Kyrre Glette, Tao Chen, Rami Bahsoon, Jim Torresen, and Xin Yao. "Architectural Aspects of Self-Aware and Self-Expressive Computing Systems: From Psychology to Engineering". In: *Computer* 48.8 (Aug. 2015), pp. 62–70. ISSN: 0018-9162.

[143] Peter R. Lewis, Marco Platzner, Bernhard Rinner, Jim Tørresen, and Xin Yao. *Self-aware Computing Systems: An Engineering Approach*. 1st ed. Springer Cham, 2016. ISBN: 978-3-319-39675-0.

[144] Chenzhao Li, Sankaran Mahadevan, You Ling, Sergio Choze, and Liping Wang. "Dynamic Bayesian Network for Aircraft Wing Health Monitoring Digital Twin". In: *AIAA Journal* 55.3 (2017), pp. 930–941. DOI: 10.2514/1.J055201.

[145] Han Li, Guoxin Wang, Jinzhi Lu, and Dimitris Kiritsis. "Cognitive twin construction for system of systems operation based on semantic integration and high-level architecture". In: *Integrated Computer-Aided Engineering* 29.3 (Jan. 2022), pp. 277–295. ISSN: 1069-2509. DOI: 10.3233/ICA-220677.

[146] Li Li, Xiao Wang, Kunfeng Wang, Yilun Lin, Jingmin Xin, Long Chen, Linhai Xu, Bin Tian, Yunfeng Ai, Jian Wang, Dongpu Cao, Yuehu Liu, Chenghong Wang, Nanning Zheng, and Fei-Yue Wang. "Parallel testing of vehicle intelligence via virtual-real interaction". In: *Science Robotics* 4.28 (2019), eaaw4106. DOI: 10.1126/scirobotics.aaw4106.

[147] Nianyu Li, Javier Cámara, David Garlan, and Bradley Schmerl. "Reasoning about When to Provide Explanation for Human-involved Self-Adaptive Systems". In: *2020 IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS)*. 2020, pp. 195–204. DOI: 10.1109/ACSOS49614.2020.00042.

[148] Xin Li, Bin He, Zhipeng Wang, Yanmin Zhou, Gang Li, and Rong Jiang. "Semantic-Enhanced Digital Twin System for Robot–Environment Interaction Monitoring". In: *IEEE Transactions on Instrumentation and Measurement* 70 (2021), pp. 1–13. ISSN: 1557-9662. DOI: 10.1109/TIM.2021.3066542.

[149] Kendrik Yan Hong Lim, Pai Zheng, and Chun-Hsien Chen. "A state-of-the-art survey of Digital Twin: techniques, engineering product lifecycle management and business

innovation perspectives". In: *Journal of Intelligent Manufacturing* (Nov. 2019). ISSN: 0956-5515. DOI: 10.1007/s10845-019-01512-w.

[150] Mengnan Liu, Shuiliang Fang, Huiyue Dong, and Cunzhi Xu. "Review of digital twin about concepts, technologies, and industrial applications". In: *Journal of Manufacturing Systems* 58 (Jan. 2021), pp. 346–361. ISSN: 02786125. DOI: 10.1016/j.jmsy.2020.06.017.

[151] Quan Liu, Zhihao Liu, Wenjun Xu, Quan Tang, Zude Zhou, and Duc Truong Pham. "Human-robot collaboration in disassembly for sustainable manufacturing". In: *International Journal of Production Research* 57.12 (June 2019), pp. 4027–4044. ISSN: 0020-7543. DOI: 10.1080/00207543.2019.1578906.

[152] Shimin Liu, Pai Zheng, Liqiao Xia, and Jinsong Bao. "A dynamic updating method of digital twin knowledge model based on fused memorizing-forgetting model". In: *Advanced Engineering Informatics* 57 (2023), p. 102115. ISSN: 1474-0346. DOI: 10.1016/j.aei.2023.102115.

[153] B. Logan and G. Theodoropoulos. "Dynamic Interest Management in the Distributed Simulation of Agent-Based Systems". In: *Proceedings of 10th AI, Simulation and Planning In High Autonomy Systems Conference*. Mar. 2000, pp. 45–50. ISBN: 1-56555-194-X.

[154] George Loukas, Tuan Vuong, Ryan Heartfield, Georgia Sakellari, Yongpil Yoon, and Diane Gan. "Cloud-Based Cyber-Physical Intrusion Detection for Vehicles Using Deep Learning". In: *IEEE Access* 6 (2018), pp. 3491–3508. DOI: 10.1109/ACCESS.2017.2782159.

[155] Giovanni Lugaresi, Sofia Gangemi, Giulia Gazzoni, and Andrea Matta. "Online Validation of Simulation-Based Digital Twins Exploiting Time Series Analysis". In: *Proceedings of the 2022 Winter Simulation Conference*. Dec. 2022.

[156] Gill Lumer-Klabbers, Jacob Odgaard Hausted, Jakob Levisen Kvistgaard, Hugo Daniel Macedo, Mirgita Frasheri, and Peter Gorm Larsen. "Towards a Digital Twin Framework for Autonomous Robots". In: *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*. July 2021, pp. 1254–1259. DOI: 10.1109/COMPSAC51774.2021.00174.

[157] Scott M. Lundberg and Su-In Lee. "A unified approach to interpreting model predictions". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., Dec. 2017, pp. 4768–4777. ISBN: 978-1-5108-6096-4. URL: https://proceedings.neurips.cc/paper/2017/hash/8a20a8621978632d76c43dfd28b67767-Abstract.html.

[158] Jianhao Lv, Xinyu Li, Yicheng Sun, Yu Zheng, and Jinsong Bao. "A bio-inspired LIDA cognitive-based digital twin architecture for unmanned maintenance of machine tools". In: *Robotics and Computer-Integrated Manufacturing* 80 (Apr. 2023), p. 102489. ISSN: 0736-5845. DOI: 10.1016/j.rcim.2022.102489.

[159] Linh Thao Ly, Fabrizio Maria Maggi, Marco Montali, Stefanie Rinderle-Ma, and Wil M. P. van der Aalst. "Compliance monitoring in business processes: Functionalities, application, and tool-support". In: *Information Systems* 54 (Dec. 2015), pp. 209–234. ISSN: 0306-4379. DOI: 10.1016/j.is.2015.02.007.

[160] Linh Thao Ly, Stefanie Rinderle-Ma, David Knuplesch, and Peter Dadam. "Monitoring Business Process Compliance Using Compliance Rule Graphs". In: *2011th Confederated International Conference on On the Move to Meaningful Internet Systems*. OTM'11. 2011, pp. 82–99. ISBN: 978-3-642-25109-2. DOI: 10.1007/978-3-642-25109-2_7.

[161] Pratik Maheshwari, Sachin Kamble, Satish Kumar, Amine Belhadi, and Shivam Gupta. "Digital twin-based warehouse management system: a theoretical toolbox for future

research and applications". In: *The International Journal of Logistics Management* (July 2023). ISSN: 0957-4093. DOI: 10.1108/IJLM-01-2023-0030.

[162] Sarah Malik, Rakeen Rouf, Krzysztof Mazur, and Antonios Kontsos. "A Dynamic Data Driven Applications Systems (DDDAS)-Based Digital Twin IoT Framework". In: *Dynamic Data Driven Applications Systems*. Ed. by Frederica Darema, Erik Blasch, Sai Ravela, and Alex Aved. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, pp. 29–36. ISBN: 978-3-030-61725-7. DOI: 10.1007/978-3-030-61725-7_6.

[163] Stefano Mariani, Marco Picone, and Alessandro Ricci. "Agents and Digital Twins for the engineering of Cyber-Physical Systems: opportunities, and challenges". In: *Annals of Mathematics and Artificial Intelligence* (July 2023). ISSN: 1573-7470. DOI: 10.1007/s10472-023-09884-9.

[164] Zafar Masood, Zheng Jiangbin, Idrees Ahmad, Chai Dongdong, Wasif Shabbir, and Muhammad Irfan. "A novel continual reinforcement learning-based expert system for self-optimization of soft real-time systems". In: *Expert Systems with Applications* 238 (Mar. 2024), p. 122309. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2023.122309.

[165] Aaron K. Massey, Richard L. Rutledge, Annie I. Antón, and Peter P. Swire. "Identifying and classifying ambiguity for regulatory requirements". In: *2014 IEEE 22nd International Requirements Engineering Conference (RE)*. Aug. 2014, pp. 83–92. DOI: 10.1109/RE.2014.6912250.

[166] R. Ryan McCune and Greg R. Madey. "Control of Artificial Swarms with DDDAS". In: *Procedia Computer Science* 29 (2014), pp. 1171–1181. ISSN: 1877-0509. DOI: 10.1016/j.procs.2014.05.105.

[167] Andreas Metzger, Clément Quinton, Zoltán Ádám Mann, Luciano Baresi, and Klaus Pohl. "Realizing self-adaptive systems via online reinforcement learning and feature-

model-guided exploration". en. In: *Computing* 106.4 (Apr. 2024), pp. 1251–1272. ISSN: 1436-5057. DOI: 10.1007/s00607-022-01052-x.

[168] Bart Meyers, Johan Van Noten, Pieter Lietaert, Bavo Tielemans, Hristo Hristov, Davy Maes, and Klaas Gadeyne. "Knowledge Graphs in Digital Twins for Manufacturing - Lessons Learned from an Industrial Case at Atlas Copco Airpower". In: *IFAC-PapersOnLine* 55.10 (2022). 10th IFAC Conference on Manufacturing Modelling, Management and Control MIM 2022, pp. 13–18. ISSN: 2405-8963. DOI: 10.1016/j.ifacol.2022.09.361.

[169] Judith Michael, Maike Schwammberger, and Andreas Wortmann. "Explaining cyber-physical system behavior with digital twins". In: *IEEE Software* 41.1 (Jan. 2024), pp. 55–63. ISSN: 1937-4194. DOI: 10.1109/MS.2023.3319580. (Visited on 04/09/2024).

[170] Fabien Michel, Jacques Ferber, and Alexis Drogoul. "Multi-Agent Systems and Simulation: A Survey from the Agent Community's Perspective". In: *Multi-Agent Systems: Simulation and Applications*. Ed. by Adelinde M. Uhrmacher and Danny Weyns. Boca Raton, FL, USA: CRC Press, 2009, pp. 3–51.

[171] Stefan Mihai, Mahnoor Yaqoob, Dang V. Hung, William Davis, Praveer Towakel, Mohsin Raza, Mehmet Karamanoglu, Balbir Barn, Dattaprasad Shetve, Raja V. Prasad, Hrishikesh Venkataraman, Ramona Trestian, and Huan X. Nguyen. "Digital Twins: A Survey on Enabling Technologies, Challenges, Trends and Future Prospects". In: *IEEE Communications Surveys & Tutorials* 24.4 (2022), pp. 2255–2291. ISSN: 1553-877X. DOI: 10.1109/COMST.2022.3208773.

[172] Tim Miller. "Explanation in Artificial Intelligence: Insights from the social sciences". In: *Artificial Intelligence* 267 (2019), pp. 1–38. ISSN: 0004-3702. DOI: 10.1016/j.artint.2018.07.007.

[173] Roberto Minerva, Gyu Myoung Lee, and Noël Crespi. "Digital Twin in the IoT Context: A Survey on Technical Features, Scenarios, and Architectural Models". In: *Pro-*

ceedings of the IEEE 108.10 (2020), pp. 1785–1824. DOI: 10.1109/JPROC.2020.2998530.

[174] Henry Muccini, Mohammad Sharaf, and Danny Weyns. "Self-adaptation for Cyber-physical Systems: A Systematic Literature Review". In: *Proceedings of the 11th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. 2016, pp. 75–81. ISBN: 978-1-4503-4187-5. DOI: 10.1145/2897053.2897069.

[175] Manuel Müller, Tamás Ruppert, Nasser Jazdi, and Michael Weyrich. "Self-improving situation awareness for human–robot-collaboration using intelligent Digital Twin". In: *Journal of Intelligent Manufacturing* (May 2023). ISSN: 1572-8145. DOI: 10.1007/s10845-023-02138-9.

[176] Manuel S. Müller, Nasser Jazdi, and Michael Weyrich. "Self-improving Models for the Intelligent Digital Twin: Towards Closing the Reality-to-Simulation Gap". In: *IFAC-PapersOnLine*. 14th IFAC Workshop on Intelligent Manufacturing Systems IMS 2022 55.2 (Jan. 2022), pp. 126–131. ISSN: 2405-8963. DOI: 10.1016/j.ifacol.2022.04.181.

[177] A. M. Mustapha, O. T. Arogundade, Sanjay Misra, Robertas Damasevicius, and Rytis Maskeliunas. "A Systematic Literature Review on Compliance Requirements Management of Business Processes". In: *International Journal of System Assurance Engineering and Management* 11.3 (June 2020), pp. 561–576. ISSN: 0976-4348. DOI: 10.1007/s13198-020-00985-w.

[178] J. Mylopoulos, L. Chung, and B. Nixon. "Representing and using nonfunctional requirements: a process-oriented approach". In: *IEEE Transactions on Software Engineering* 18.6 (1992), pp. 483–497. DOI: 10.1109/32.142871.

[179] Htet Naing, Wentong Cai, Nan Hu, Tiantian Wu, and Liang Yu. "Data-Driven Microscopic Traffic Modelling and Simulation Using Dynamic LSTM". In: *Proceedings of the 2021 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*. New York, NY, USA: ACM, 2021, pp. 1–12. ISBN: 978-1-4503-8296-0.

[180] Claudia Negri-Ribalta, René Noel, Nicolas Herbaut, Oscar Pastor, and Camille Salinesi. "Socio-Technical Modelling for GDPR Principles: an Extension for the STS-ml". In: *2022 IEEE 30th International Requirements Engineering Conference Workshops (REW)*. Aug. 2022, pp. 238–234. DOI: 10.1109/REW56159.2022.00052.

[181] Subash Neupane, Ivan A. Fernandez, Wilson Patterson, Sudip Mittal, Milan Parmar, and Shahram Rahimi. *TwinExplainer: Explaining predictions of an automotive digital twin*. Jan. 2023. DOI: 10.48550/arXiv.2302.00152.

[182] Sergei Nikolaev, Peter D. Barnes, James M. Brase, Thomas W. Canales, David R. Jefferson, Steve Smith, Ron A. Soltz, and Peter J. Scheibel. "Performance of distributed ns-3 network simulator". In: *Proceedings of the 6th International ICST Conference on Simulation Tools and Techniques*. SimuTools '13. Brussels, BEL, Mar. 2013, pp. 17–23. ISBN: 978-1-4503-2464-9.

[183] Shin-ya Nishizaki and Takuya Ohata. "Real-Time Model Checking for Regulatory Compliance". In: *International Conference on Advances in Information Technology and Mobile Communication*. 2013, pp. 70–77. ISBN: 978-3-642-35864-7. DOI: 10.1007/978-3-642-35864-7_10.

[184] Michael J North, Nicholson T Collier, Jonathan Ozik, Eric R Tatara, Charles M Macal, Mark Bragen, and Pam Sydelko. "Complex Adaptive Systems Modeling with Repast Simphony". In: *Complex Adaptive Systems Modeling* 1.1 (2013), p. 3. ISSN: 2194-3206.

[185] Rotimi Ogunsakin, Nikolay Mehandjiev, and Cesar A. Marin. "Towards adaptive digital twins architecture". In: *Computers in Industry* 149 (Aug. 2023), p. 103920. ISSN: 0166-3615. DOI: 10.1016/j.compind.2023.103920.

[186] Bendra Ojameruaye and Rami Bahsoon. "Systematic Elaboration of Compliance Requirements Using Compliance Debt and Portfolio Theory". In: *20th International Working Conference on Requirements Engineering: Foundation for Software Quality*

(REFSQ). 2014, pp. 152–167. ISBN: 978-3-319-05843-6. DOI: 10.1007/978-3-319-05843-6_12.

[187] Bhakti Stephan Onggo, Navonil Mustafee, Andi Smart, Angel A. Juan, and Owen Molloy. "Symbiotic Simulation System: Hybrid Systems Model Meets Big Data Analytics". In: *2018 Winter Simulation Conference (WSC)*. Vol. 2018-Decem. 2017. IEEE, Dec. 2018, pp. 1358–1369. ISBN: 978-1-5386-6572-5. DOI: 10.1109/WSC.2018.8632407.

[188] Paul N. Otto and Annie I. Anton. "Addressing Legal Requirements in Requirements Engineering". In: *15th IEEE International Requirements Engineering Conference (RE 2007)*. Oct. 2007, pp. 5–14. DOI: 10.1109/RE.2007.65.

[189] Alexander Palm, Andreas Metzger, and Klaus Pohl. "Online Reinforcement Learning for Self-adaptive Information Systems". In: *Advanced Information Systems Engineering*. Ed. by Schahram Dustdar, Eric Yu, Camille Salinesi, Dominique Rieu, and Vik Pant. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, pp. 169–184. ISBN: 978-3-030-49435-3. DOI: 10.1007/978-3-030-49435-3_11.

[190] Vasileia Papathanasopoulou, Ioulia Markou, and Constantinos Antoniou. "Online Calibration for Microscopic Traffic simulation and Dynamic Multi-Step Prediction of Traffic Speed". In: *Transportation Research Part C: Emerging Technologies* 68 (July 2016), pp. 144–159. ISSN: 0968-090X.

[191] Ken Peffers, Tuure Tuunanen, Marcus A. Rothenberger, and Samir Chatterjee. "A design science research methodology for information systems research". In: *Journal of Management Information Systems* 24.3 (Dec. 2007), pp. 45–77. ISSN: 0742-1222. DOI: 10.2753/MIS0742-1222240302.

[192] Jérôme Pfeiffer, Daniel Lehner, Andreas Wortmann, and Manuel Wimmer. "Towards a Product Line Architecture for Digital Twins". In: *2023 IEEE 20th International Conference on Software Architecture Companion (ICSA-C)*. ISSN: 2768-4288. Mar. 2023, pp. 187–190. DOI: 10.1109/ICSA-C57050.2023.00049.

[193] Danilo Pianini, Federico Pettinari, Roberto Casadei, and Lukas Esterle. "A Collective Adaptive Approach to Decentralised K-Coverage in Multi-Robot Systems". In: *ACM Trans. Auton. Adapt. Syst.* 17.1–2 (Sept. 2022). ISSN: 1556-4665. DOI: 10.1145/3547145.

[194] Federico Quin, Danny Weyns, and Omid Gheibi. "Decentralized Self-Adaptive Systems: A Mapping Study". In: *2021 International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. 2021, pp. 18–29. DOI: 10.1109/SEAMS51251.2021.00014.

[195] Radhakisan Baheti and Helen Gill. "Cyber-physical Systems". In: *The Impact of Control Technology* (2011), pp. 1–6. URL: https://ieeecss.org/sites/ieeecss/files/2019-07/IoCT-Part3-02CyberphysicalSystems.pdf.

[196] Adil Rasheed, Omer San, and Trond Kvamsdal. "Digital Twin: Values, Challenges and Enablers From a Modeling Perspective". In: *IEEE Access* 8 (2020), pp. 21980–22012. DOI: 10.1109/ACCESS.2020.2970143.

[197] M. Mazhar Rathore, Syed Attique Shah, Dhirendra Shukla, Elmahdi Bentafat, and Spiridon Bakiras. "The Role of AI, Machine Learning, and Big Data in Digital Twinning: A Systematic Literature Review, Challenges, and Opportunities". In: *IEEE Access* 9 (2021), pp. 32030–32052. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2021.3060863.

[198] Deepti Balaji Raykar, L. T. JayPrakash, and K. V. Dinesha. "An Iterative and Incremental Approach to Address Regulatory Compliance Concerns in Requirements Engineering". In: *10th International Advanced Computing Conference (IACC)*. 2021, pp. 323–335. ISBN: 9789811604041. DOI: 10.1007/978-981-16-0404-1_24.

[199] Alessandro Ricci, Angelo Croatti, Stefano Mariani, Sara Montagna, and Marco Picone. "Web of Digital Twins". In: *ACM Transactions on Internet Technology* 22.4 (Nov. 2022), 101:1–101:30. ISSN: 1533-5399. DOI: 10.1145/3507909.

[200] Bernhard Rinner, Lukas Esterle, Jennifer Simonjan, Georg Nebehay, Roman Pflugfelder, Gustavo Fernández Domínguez, and Peter R. Lewis. "Self-Aware and Self-Expressive Camera Networks". In: *Computer* 48.7 (2015), pp. 21–28. DOI: 10.1109/MC.2015.209.

[201] T.G. Ritto and F.A. Rochinha. "Digital twin, physics-based model, and machine learning applied to damage detection in structures". In: *Mechanical Systems and Signal Processing* 155 (June 2021), p. 107614. ISSN: 08883270. DOI: 10.1016/j.ymssp.2021.107614.

[202] Luis F Rivera, Miguel Jiménez, Prashanti Angara, Norha M Villegas, Gabriel Tamura, and Hausi A Müller. "Towards Continuous Monitoring in Personalized Healthcare Through Digital Twins". In: *Proceedings of the 29th Annual International Conference on Computer Science and Software Engineering*. 2019, pp. 329–335. ISBN: 978-1-4503-9999-9. URL: https://dl.acm.org/doi/10.5555/3370272.3370310.

[203] Luis F Rivera, Hausi A Müller, Norha M Villegas, Gabriel Tamura, and Miguel Jiménez. "On the Engineering of IoT-Intensive Digital Twin Software Systems". In: *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*. ICSEW'20. New York, NY, USA: ACM, 2020, pp. 631–638. ISBN: 9781450379632.

[204] Luis F. Rivera, Miguel Jimenez, Norha M. Villegas, Gabriel Tamura, and Hausi A. Muller. "Toward Autonomic, Software-Intensive Digital Twin Systems". In: *IEEE Software* (2021). Conference Name: IEEE Software, pp. 20–26. ISSN: 1937-4194. DOI: 10.1109/MS.2021.3133913.

[205] Luis F. Rivera, Miguel Jiménez, Norha M. Villegas, Gabriel Tamura, and Hausi A. Müller. "The Forging of Autonomic and Cooperating Digital Twins". In: *IEEE Internet Computing* 26.5 (2022), pp. 41–49. DOI: 10.1109/MIC.2021.3051902.

[206] Sujit Rokka Chhetri and Mohammad Abdullah Al Faruque. "Dynamic Data-Driven Digital Twin Modeling". In: *Data-Driven Modeling of Cyber-Physical Systems using*

Side-Channel Analysis. Cham: Springer International Publishing, 2020, pp. 129–153. ISBN: 978-3-030-37962-9. DOI: 10.1007/978-3-030-37962-9_7.

[207] Jürgen Roßmann, Eric Guiffo Kaigom, Linus Atorf, Malte Rast, Georgij Grinshpun, and Christian Schlette. "Mental Models for Intelligent Systems: eRobotics Enables New Approaches to Simulation-Based AI". In: *KI - Künstliche Intelligenz* 28.2 (June 2014), pp. 101–110. DOI: 10.1007/s13218-014-0298-z.

[208] Jože M. Rožanec, Jinzhi Lu, Jan Rupnik, Maja Škrjanc, Dunja Mladenić, Blaž Fortuna, Xiaochen Zheng, and Dimitris Kiritsis. "Actionable cognitive twins for decision making in manufacturing". In: *International Journal of Production Research* 60.2 (Jan. 2022), pp. 452–478. ISSN: 0020-7543. DOI: 10.1080/00207543.2021.2002967.

[209] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. 3rd. USA: Prentice Hall Press, 2009. ISBN: 0136042597.

[210] Nada Sahlab, Simon Kamm, Timo Müller, Nasser Jazdi, and Michael Weyrich. "Knowledge Graphs as Enhancers of Intelligent Digital Twins". In: *2021 4th IEEE International Conference on Industrial Cyber-Physical Systems (ICPS)*. May 2021, pp. 19–24. DOI: 10.1109/ICPS49255.2021.9468219.

[211] Abhishek Sainani, Preethu Rose Anish, Vivek Joshi, and Smita Ghaisas. "Extracting and Classifying Requirements from Software Engineering Contracts". In: *2020 IEEE 28th International Requirements Engineering Conference (RE)*. Aug. 2020, pp. 147–157. DOI: 10.1109/RE48521.2020.00026.

[212] Mazeiar Salehie and Ladan Tahvildari. "Self-Adaptive Software: Landscape and Research Challenges". In: *ACM Transactions on Autonomous and Adaptive Systems* 4.2 (May 2009), pp. 1–42. ISSN: 1556-4665. DOI: 10.1145/1516533.1516538.

[213] César Sánchez, Gerardo Schneider, Wolfgang Ahrendt, Ezio Bartocci, Domenico Bianculli, Christian Colombo, Yliès Falcone, Adrian Francalanza, Srđan Krstić, Joao M.

Lourenço, Dejan Nickovic, Gordon J. Pace, Jose Rufino, Julien Signoles, Dmitriy Traytel, and Alexander Weiss. "A survey of challenges for runtime verification from advanced application domains (beyond software)". In: *Formal Methods in System Design* 54.3 (Nov. 2019), pp. 279–335. ISSN: 1572-8102. DOI: 10.1007/s10703-019-00337-w.

[214] Carlos Henrique dos Santos, José Arnaldo Barra Montevechi, José Antônio de Queiroz, Rafael de Carvalho Miranda, and Fabiano Leal. "Decision support in productive processes through DES and ABS in the Digital Twin era: a systematic literature review". In: *International Journal of Production Research* (2021), pp. 1–20. ISSN: 1366588X.

[215] Robert G. Sargent. "Verification and validation of simulation models". In: *Proceedings of the 2010 Winter Simulation Conference*. 2010, pp. 166–183. DOI: 10.1109/WSC.2010.5679166.

[216] Sara Sartoli, Sepideh Ghanavati, and Akbar Siami Namin. "Compliance Requirements Checking in Variable Environments". In: *2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC)*. July 2020, pp. 1093–1094. DOI: 10.1109/COMPSAC48688.2020.0-124.

[217] A.J. van der Schaft. "Equivalence of Dynamical Systems by Bisimulation". In: *IEEE Transactions on Automatic Control* 49.12 (Dec. 2004), pp. 2160–2172. ISSN: 1558-2523.

[218] Julian Schrittwieser, Thomas Hubert, Amol Mandhane, Mohammadamin Barekatain, Ioannis Antonoglou, and David Silver. "Online and Offline Reinforcement Learning by Planning with a Learned Model". In: *Advances in Neural Information Processing Systems*. Vol. 34. Curran Associates, Inc., 2021, pp. 27580–27591. URL: https://proceedings.neurips.cc/paper/2021/hash/e8258e5140317ff36c7f8225a3bf9590-Abstract.html.

[219] Angira Sharma, Edward Kosasih, Jie Zhang, Alexandra Brintrup, and Anisoara Calinescu. “Digital Twins: State of the art theory and practice, challenges, and open research questions”. In: *Journal of Industrial Information Integration* 30 (Nov. 2022), p. 100383. ISSN: 2452-414X. DOI: 10.1016/j.jii.2022.100383.

[220] Rahul Sharma, Eric Schkufza, Berkeley Churchill, and Alex Aiken. “Data-Driven Equivalence Checking”. In: *"Proceedings of the 2013 ACM SIGPLAN International Conference on Object Oriented Programming Systems Languages & Applications"*. OOPSLA ’13. Indianapolis, Indiana, USA: ACM, 2013, pp. 391–406. ISBN: 9781450323741.

[221] Gaoqing Shen, Lei Lei, Zhilin Li, Shengsuo Cai, Lijuan Zhang, Pan Cao, and Xiaojiao Liu. “Deep Reinforcement Learning for Flocking Motion of Multi-UAV Systems: Learn From a Digital Twin”. In: *IEEE Internet of Things Journal* 9.13 (July 2022), pp. 11141–11153. ISSN: 2327-4662, 2372-2541. DOI: 10.1109/JIOT.2021.3127873.

[222] He Shen, Alexis Ruiz, and Ni Li. “Fast online reinforcement learning control of small lift-driven vertical axis wind turbines with an active programmable four bar linkage mechanism”. In: *Energy* 262 (2023), p. 125350. ISSN: 0360-5442. DOI: 10.1016/j.energy.2022.125350.

[223] Sarah Stieß, Steffen Becker, Florian Ege, Stefan Höppner, and Matthias Tichy. “Coordination and explanation of reconfigurations in self-adaptive high-performance systems”. In: *MODELS ’22: ACM/IEEE 25th International Conference on Model Driven Engineering Languages and Systems*. MODELS ’22. New York, NY, USA: Association for Computing Machinery, Oct. 2022, pp. 486–490. ISBN: 978-1-4503-9467-3. DOI: 10.1145/3550356.3561555.

[224] Nenad Stojanovic and Dejan Milenovic. “Data-driven Digital Twin approach for process optimization: an industry use case”. In: *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, Dec. 2018, pp. 4202–4211. ISBN: 978-1-5386-5035-6. DOI: 10.1109/BigData.2018.8622412.

[225] Wen Sun, Shiyu Lei, Lu Wang, Zhiqiang Liu, and Yan Zhang. "Adaptive Federated Learning and Digital Twin for Industrial Internet of Things". In: *IEEE Transactions on Industrial Informatics* 17.8 (Aug. 2021), pp. 5605–5614. ISSN: 1941-0050. DOI: 10.1109/TII.2020.3034674.

[226] Vinoth Suryanarayanan, Bart G. W. Craenen, and Georgios K. Theodoropoulos. "Synchronised Range Queries in Distributed Simulations of Multi-agent Systems". In: *2010 IEEE/ACM 14th International Symposium on Distributed Simulation and Real Time Applications*. 2010, pp. 79–86. DOI: 10.1109/DS-RT.2010.18.

[227] Vinoth Suryanarayanan and Georgios Theodoropoulos. "Synchronised Range Queries in Distributed Simulations of Multiagent Systems". In: *ACM Trans. Model. Comput. Simul.* 23.4 (Nov. 2013). ISSN: 1049-3301.

[228] Barıs Tan and Andrea Matta. "Optimizing Digital Twin Synchronization in a Finite Horizon". In: *Proceedings of the 2022 Winter Simulation Conference*. Dec. 2022.

[229] Barış Tan and Andrea Matta. "The Digital Twin Synchronization Problem: Framework, Formulations, and Analysis". In: *IISE Transactions* (Aug. 2023), pp. 1–25. ISSN: 2472-5854. DOI: 10.1080/24725854.2023.2253869.

[230] Fei Tao, Fangyuan Sui, Ang Liu, Qinglin Qi, Meng Zhang, Boyang Song, Zirong Guo, Stephen C.Y. Lu, and A. Y. C. Nee. "Digital twin-driven product design framework". In: *International Journal of Production Research* 57.12 (June 2019), pp. 3935–3953. ISSN: 0020-7543. DOI: 10.1080/00207543.2018.1443229.

[231] Fei Tao, He Zhang, Ang Liu, and A. Y.C. C Nee. "Digital Twin in Industry: State-of-the-Art". In: *IEEE Transactions on Industrial Informatics* 15.4 (Apr. 2019), pp. 2405–2415. ISSN: 1551-3203. DOI: 10.1109/TII.2018.2873186.

[232] Fei Tao and Meng Zhang. "Digital Twin Shop-Floor: A New Shop-Floor Paradigm Towards Smart Manufacturing". In: *IEEE Access* 5 (2017), pp. 20418–20427. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2017.2756069.

[233] Adam Thelen, Xiaoge Zhang, Olga Fink, Yan Lu, Sayan Ghosh, Byeng D. Youn, Michael D. Todd, Sankaran Mahadevan, Chao Hu, and Zhen Hu. "A comprehensive review of digital twin — part 1: modeling and twinning enabling technologies". In: *Structural and Multidisciplinary Optimization* 65.12 (Nov. 2022), p. 55. ISSN: 1615-1488. DOI: 10.1007/s00158-022-03425-4.

[234] US Department of Defense. *DoD Modeling and Simulation (M&S) Verification, Validation, and Accreditation (VV&A): DoD Instruction 5000.61*. Tech. rep. US Department of Defense, May 2009.

[235] Hendrik van der Valk, Hendrik Hasse, Frederik Moeller, Michael Arbter, Jan-Luca Henning, and Boris Otto. "A Taxonomy of Digital Twins". In: *AMCIS 2020 Proceedings, Americas Conference on Information Systems*. 2020. ISBN: 978-1-73363-254-6. URL: https://aisel.aisnet.org/amcis2020/org_transformation_is/org_transformation_is/4.

[236] Raymon van Dinter, Bedir Tekinerdogan, and Cagatay Catal. "Predictive maintenance using digital twins: A systematic literature review". In: *Information and Software Technology* 151 (2022), p. 107008. ISSN: 0950-5849. DOI: 10.1016/j.infsof.2022.107008.

[237] Eric VanDerHorn and Sankaran Mahadevan. "Digital Twin: Generalization, characterization and implementation". In: *Decision Support Systems* 145 (June 2021), p. 113524. ISSN: 0167-9236. DOI: 10.1016/j.dss.2021.113524.

[238] Michael Vierhauser, Rick Rabiser, and Paul Grünbacher. "Requirements monitoring frameworks: A systematic review". In: *Information and Software Technology* 80 (Dec. 2016), pp. 89–109. ISSN: 0950-5849. DOI: 10.1016/j.infsof.2016.08.005.

[239] João L. Vilar-Dias, Adelson Santos S. Junior, and Fernando B. Lima-Neto. "An interpretable digital twin for self-aware industrial machines". en. In: *Sensors* 24.1 (Jan. 2024), p. 27. ISSN: 1424-8220. DOI: 10.3390/s24010004. (Visited on 04/09/2024).

[240] Andrew Wagenmaker and Aldo Pacchiano. "Leveraging offline data in online reinforcement learning". In: *Proceedings of the 40th International Conference on Machine Learning*. PMLR, July 2023, pp. 35300–35338. URL: https://proceedings.mlr.press/v202/wagenmaker23a.html.

[241] Chi-Hsu Wang and Jung-Sheng Wen. "On the Equivalence of a Table Lookup (TL) Technique and Fuzzy Neural Network (FNN) With Block Pulse Membership Functions (BPMFs) and Its Application to Water Injection Control of an Automobile". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 38.4 (July 2008), pp. 574–580. ISSN: 1558-2442.

[242] Fei-Yue Wang. "Parallel Control and Management for Intelligent Transportation Systems: Concepts, Architectures, and Applications". In: *IEEE Transactions on Intelligent Transportation Systems* 11.3 (Sept. 2010). Conference Name: IEEE Transactions on Intelligent Transportation Systems, pp. 630–638. ISSN: 1558-0016. DOI: 10.1109/TITS.2010.2060218.

[243] Fei-Yue Wang. "Parallel system methods for management and control of complex systems". In: *Control and Decision* 19.5 (May 2004), pp. 485–489.

[244] Fei-Yue Wang, Xiao Wang, Lingxi Li, and Li Li. "Steps toward Parallel Intelligence". In: *IEEE/CAA Journal of Automatica Sinica* 3.4 (2016), pp. 345–348. DOI: 10.1109/JAS.2016.7510067.

[245] Haozhe Wang, Yulei Wu, Geyong Min, and Wang Miao. "A Graph Neural Network-Based Digital Twin for Network Slicing Management". In: *IEEE Transactions on Industrial Informatics* 18.2 (Feb. 2022), pp. 1367–1376. ISSN: 1941-0050. DOI: 10.1109/TII.2020.3047843.

[246] Lu Wang, Tianhu Deng, Zeyu Zheng, and Zuo-Jun Max Shen. "Explainable Modeling in Digital Twin". In: *Proceedings of the Winter Simulation Conference*. WSC '21. IEEE Press, 2021.

[247] Shuai Wang, Jing Wang, Xiao Wang, Tianyu Qiu, Yong Yuan, Liwei Ouyang, Yuanyuan Guo, and Fei-Yue Wang. "Blockchain-Powered Parallel Healthcare Systems Based on the ACP Approach". In: *IEEE Transactions on Computational Social Systems* 5.4 (2018), pp. 942–950. DOI: 10.1109/TCSS.2018.2865526.

[248] Xiao Wang, Jing Yang, Jinpeng Han, Wei Wang, and Fei-Yue Wang. "Metaverses and DeMetaverses: From Digital Twins in CPS to Parallel Intelligence in CPSS". In: *IEEE Intelligent Systems* 37.4 (2022), pp. 97–102. DOI: 10.1109/MIS.2022.3196592.

[249] Xiucheng Wang, Longfei Ma, Haocheng Li, Zhisheng Yin, Tom. Luan, and Nan Cheng. "Digital Twin-Assisted Efficient Reinforcement Learning for Edge Task Scheduling". In: *2022 IEEE 95th Vehicular Technology Conference: (VTC2022-Spring)*. June 2022, pp. 1–5. DOI: 10.1109/VTC2022-Spring54318.2022.9860495.

[250] Yuepeng Wang, Isil Dillig, Shuvendu K. Lahiri, and William R. Cook. "Verifying equivalence of database-driven applications". In: *Proceedings of the ACM on Programming Languages* 2.POPL (Jan. 2018), pp. 1–29. ISSN: 2475-1421. DOI: 10.1145/3158144.

[251] Yuntao Wang, Zhou Su, Shaolong Guo, Minghui Dai, Tom H. Luan, and Yiliang Liu. "A Survey on Digital Twins: Architecture, Enabling Technologies, Security and Privacy, and Future Prospects". In: *IEEE Internet of Things Journal* (Apr. 2023), pp. 1–21. ISSN: 2327-4662. DOI: 10.1109/JIOT.2023.3263909.

[252] Maryna Waszak, An Ngoc Lam, Volker Hoffmann, Brian Elvesæter, Maria Flavia Mogos, and Dumitru Roman. "Let the Asset Decide: Digital Twins with Knowledge Graphs". In: *2022 IEEE 19th International Conference on Software Architecture*

Companion (ICSA-C). ISSN: 2768-4288. Mar. 2022, pp. 35–39. DOI: 10.1109/ICSA-C54293.2022.00014.

[253] Geoffrey I. Webb, Roy Hyde, Hong Cao, Hai Long Nguyen, and Francois Petitjean. "Characterizing concept drift". In: *Data Mining and Knowledge Discovery* 30.4 (July 2016), pp. 964–994. ISSN: 1573-756X. DOI: 10.1007/s10618-015-0448-4.

[254] Kris Welsh, Nelly Bencomo, Pete Sawyer, and Jon Whittle. "Self-Explanation in Adaptive Systems Based on Runtime Goal-Based Models". In: *Transactions on Computational Collective Intelligence XVI*. Ed. by Ryszard Kowalczyk and Ngoc Thanh Nguyen. Springer Berlin Heidelberg, 2014, pp. 122–145. ISBN: 978-3-662-44871-7. DOI: 10.1007/978-3-662-44871-7_5.

[255] Danny Weyns. *An introduction to self-adaptive systems: a contemporary software engineering perspective*. Hoboken: Wiley, 2021. ISBN: 978-1-119-57494-1.

[256] Danny Weyns. "Software engineering of self-adaptive systems". In: *Handbook of Software Engineering*. Ed. by Sungdeok Cha, Richard N. Taylor, and Kyochul Kang. Cham: Springer International Publishing, 2019, pp. 399–443. ISBN: 978-3-030-00262-6. DOI: 10.1007/978-3-030-00262-6_11.

[257] Danny Weyns, Bradley Schmerl, Vincenzo Grassi, Sam Malek, Raffaela Mirandola, Christian Prehofer, Jochen Wuttke, Jesper Andersson, Holger Giese, and Karl M Göschka. "On Patterns for Decentralized Control in Self-Adaptive Systems". In: *Software Engineering for Self-Adaptive Systems II: International Seminar, Dagstuhl Castle, Germany, October 24-29, 2010 Revised Selected and Invited Papers*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 76–107. ISBN: 978-3-642-35813-5.

[258] Lyndon While, Phil Hingston, Luigi Barone, and Simon Huband. "A Faster Algorithm for Calculating Hypervolume". In: *IEEE Transactions on Evolutionary Computation* 10.1 (2006), pp. 29–38.

[259] Michael Wooldridge and Nicholas R. Jennings. "Intelligent Agents: Theory and Practice". In: *The Knowledge Engineering Review* 10.2 (1995), pp. 115–152.

[260] Yiwen Wu, Ke Zhang, and Yan Zhang. "Digital Twin Networks: a Survey". In: *IEEE Internet of Things Journal* (2021), pp. 1–17. ISSN: 2327-4662. DOI: 10.1109/JIOT.2021.3079510.

[261] Hansong Xu, Jun Wu, Qianqian Pan, Xinping Guan, and Mohsen Guizani. "A Survey on Digital Twin for Industrial Internet of Things: Applications, Technologies and Tools". In: *IEEE Communications Surveys & Tutorials* (July 2023), pp. 1–1. ISSN: 1553-877X. DOI: 10.1109/COMST.2023.3297395.

[262] Ju Xu and Zhanxing Zhu. "Reinforced continual learning". In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. NIPS'18. Red Hook, NY, USA: Curran Associates Inc., Dec. 2018, pp. 907–916. URL: https://proceedings.neurips.cc/paper_files/paper/2018/file/cee631121c2ec9232f3a2f028ad5c89b-Paper.pdf.

[263] Lin-Yao Yang, Si-Yuan Chen, Xiao Wang, Jun Zhang, and Cheng-Hong Wang. "Digital twins and parallel systems: state of the art, comparisons and prospect". In: *Acta Automatica Sinica* 45.11 (2019), pp. 2001–2031.

[264] Senquan Yang, Fan Ding, Pu Li, and Songxi Hu. "Distributed multi-camera multi-target association for real-time tracking". In: *Scientific Reports* 12.1 (June 2022). Number: 1 Publisher: Nature Publishing Group, p. 11052. ISSN: 2045-2322. DOI: 10.1038/s41598-022-15000-4.

[265] Enes Yigitbas, Kadiray Karakaya, Ivan Jovanovikj, and Gregor Engels. "Enhancing Human-in-the-Loop Adaptive Systems through Digital Twins and VR Interfaces". In: *2021 International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. ISSN: 2157-2321. May 2021, pp. 30–40. DOI: 10.1109/SEAMS51251.2021.00015.

[266] E.S.K. Yu. "Towards modelling and reasoning support for early-phase requirements engineering". In: *Proceedings of ISRE '97: 3rd IEEE International Symposium on Requirements Engineering*. Jan. 1997, pp. 226–235. DOI: 10.1109/ISRE.1997.566873. (Visited on 04/01/2024).

[267] Luyao Yuan, Xiaofeng Gao, Zilong Zheng, Mark Edmonds, Ying Nian Wu, Federico Rossano, Hongjing Lu, Yixin Zhu, and Song-Chun Zhu. "In situ bidirectional human-robot value alignment". In: *Science Robotics* 7.68 (July 2022), eabm4183. DOI: 10.1126/scirobotics.abm4183.

[268] Lei Yue and Xiaobo Li. "A Smart Manufacturing Compliance Architecture of Electronic Batch Recording System (eBRS) for Life Sciences Industry". In: *2018 3rd International Conference on Mechanical, Control and Computer Engineering (ICMCCE)*. Sept. 2018, pp. 206–212. DOI: 10.1109/ICMCCE.2018.00050.

[269] Jiazheng Zhang, Jingkun Yan, Xiujuan Du, and Long Jin. "Multi-Robot Distributed Coordination in IOT Systems: An ACP Framework". In: *2022 IEEE 2nd International Conference on Digital Twins and Parallel Intelligence (DTPI)*. 2022, pp. 1–4. DOI: 10.1109/DTPI55838.2022.9998940.

[270] Ke Zhang, Jiayu Cao, and Yan Zhang. "Adaptive Digital Twin and Multi-agent Deep Reinforcement Learning for Vehicular Edge Computing and Networks". In: *IEEE Transactions on Industrial Informatics* 18.2 (Feb. 2022), pp. 1405–1413. ISSN: 1941-0050. DOI: 10.1109/TII.2021.3088407.

[271] Zhiheng Zhao, Mengdi Zhang, Jian Chen, Ting Qu, and George Q. Huang. "Digital twin-enabled dynamic spatial-temporal knowledge graph for production logistics resource allocation". In: *Computers & Industrial Engineering* 171 (Sept. 2022), p. 108454. ISSN: 0360-8352. DOI: 10.1016/j.cie.2022.108454.

[272] Xiaochen Zheng, Jinzhi Lu, and Dimitris Kiritsis. "The emergence of cognitive digital twin: Vision, challenges and opportunities". In: *International Journal of Production*

Research 60.24 (Dec. 2022), pp. 7610–7632. ISSN: 0020-7543, 1366-588X. DOI: 10.1080/00207543.2021.2014591.

[273] Junlong Zhou, Jin Sun, Mingyue Zhang, and Yue Ma. "Dependable Scheduling for Real-Time Workflows on Cyber–Physical Cloud Systems". In: *IEEE Transactions on Industrial Informatics* 17.11 (2021), pp. 7820–7829. DOI: 10.1109/TII.2020.3011506.

[274] Fenghua Zhu, Yisheng Lv, Yuanyuan Chen, Xiao Wang, Gang Xiong, and Fei-Yue Wang. "Parallel Transportation Systems: Toward IoT-Enabled Smart Urban Traffic Control and Management". In: *IEEE Transactions on Intelligent Transportation Systems* 21.10 (2020), pp. 4063–4071. DOI: 10.1109/TITS.2019.2934991.

[275] Q.K. Zhu. "Logic Equivalence Check in SOC Design: Solution and Issues". In: *IET Conference Proceedings*. Beijing, China: Institution of Engineering and Technology, 2008, 623–626(3).

[276] Zexuan Zhu, Chao Liu, and Xun Xu. "Visualisation of the Digital Twin data in manufacturing by using Augmented Reality". In: *Procedia CIRP* 81 (2019). 52nd CIRP Conference on Manufacturing Systems (CMS), Ljubljana, Slovenia, June 12-14, 2019, pp. 898–903. ISSN: 2212-8271. DOI: 10.1016/j.procir.2019.03.223.

[277] H Zipper and C Diedrich. "Synchronization of Industrial Plant and Digital Twin". In: *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. Zaragoza, Spain: IEEE Press, 2019, pp. 1678–1681.