Towards a Modular Deep Reinforcement Learning Digital-Twins Framework: A Step towards optimal RMS control

Abdelfatah Kermali, Madani Bezoui, Samir Ouchani, Ahcene Bounceur

▶ **To cite this version:**

Abdelfatah Kermali, Madani Bezoui, Samir Ouchani, Ahcene Bounceur. Towards a Modular Deep Reinforcement Learning Digital-Twins Framework: A Step towards optimal RMS control. The 7th International Conference on Future Networks & Distributed Systems, Dec 2023, Dubai, United Arab Emirates. hal-04389196

HAL Id: hal-04389196
https://hal.science/hal-04389196v1

Submitted on 11 Jan 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Towards a Modular Deep Reinforcement Learning Digital-Twins Framework: A Step towards optimal RMS control

Abdelfatah Kermali
CESI LINEACT, UR 7527
Nice, France
akermali@cesi.fr

Madani Bezoui
CESI LINEACT, UR 7527
Nice, France
mbezoui@cesi.fr

Samir Ouchani
CESI LINEACT, UR 7527
Aix-en-Provence, France
souchani@cesi.fr

Ahcene Bounceur
Computer Science Department
KFUPM
Dhahran, Saudi Arabia
Ahcene.Bounceur@kfupm.edu.sa

ABSTRACT

This paper proposes a modular deep reinforcement learning framework integrated with digital twin technology for optimizing the control of Reconfigurable Manufacturing Systems (RMS). The framework employs hierarchical deep reinforcement learning agents for scheduling and reconfiguration decisions across decentralized digital twins of individual Reconfigurable Machine Tools (RMT). The digital twins enable real-time monitoring, simulation, and visualization to inform the reinforcement learning agents. The modular architecture aligns with RMS goals of adaptability and rapid reconfiguration. The reconfiguration agent selects optimal machine configurations based on assessments of job queues, tardiness costs, and due dates. The scheduling agent optimizes job sequencing given the current configuration. The RMS environment coordinates the machine agents for overall optimization. Predictive maintenance capabilities are also incorporated within the digital twins. This integration of digital twins and deep reinforcement learning provides capabilities for optimal control of resilient and efficient RMS, to be responsive to dynamic manufacturing environments.

Keywords

Reconfigurable Manufacturing Systems (RMS); Deep Reinforcement Learning (DRL); Digital Twin Technology (DTT); Modern Manufacturing; Multi/Hierarchical agents Systems; RMS control; Industry 4.0

I. INTRODUCTION

The rapid pace of technological evolution and fluctuating market dynamics continue to drive the industrial and manufacturing sectors to seek innovative solutions to meet the growing demand for product customisation and unpredictable market trends [1]. One of the most notable solutions to these challenges has been the emergence of Reconfigurable Manufacturing Systems (RMS), which represent a remarkable shift in the manufacturing paradigm, combining the high-throughput capabilities of Dedicated Manufacturing Lines (DML) with the adaptability and responsiveness of Flexible Manufacturing Systems (FMS), enabling a robust response to market changes with efficiency and agility [2]. The transition from DML to FMS, and ultimately to RMS, underscores a profound change in manufacturing philosophy that prioritises

adaptability, modularity and rapid reconfiguration to overcome the dynamic hurdles of today's industrial landscapes.

A core aspect of RMS revolves around three key stages: design, implementation and optimisation. The design phase lays the foundation by defining the system architecture, which in drives the following implementation and optimisation phases. Crucial decisions about system components and possible reconfigurations are made during this phase [3]. Following the design phase, the implementation phase brings the theoretical design to life and transforms it into a functional and operational system. This phase is full of complexities, involving the integration of different system components, ensuring operational coherence and addressing practical considerations [4]. Finally, the Optimisation phase continually refines system operations to increase efficiency and productivity while reducing costs and waste. This phase employs a variety of optimisation tools and strategies, such as scheduling, machine allocation and production planning, and requires the consideration of real-time data and prospective reconfigurations [5].

Recently, the fusion of Artificial Intelligence (AI) and learning methods, in particular Reinforcement Learning (RL), has received considerable attention in solving complex problems in various domains. These technologies have the potential to improve decision making, system adaptation and optimisation, thereby addressing some of the critical challenges associated with RMS implementation.

The integration of Digital Twin Technology (DTT) represents a new frontier in improving the control and optimisation of RMS. By creating a digital replica of the physical manufacturing system, DTT facilitates real-time monitoring, analysis and control, providing a robust platform for implementing and evaluating RMS strategies. This merging of DTT and RL promotes a holistic framework that not only addresses the inherent challenges of RMS, but also drives the system to new levels of efficiency, adaptability and responsiveness.

II. Literature Review

RL's suitability for RMS results from its optimization capability through system interaction without requiring an accurate model. Initial research by Riedmiller et al. [6] showcased RL's effectiveness in acquiring dispatching policies for production scheduling. Successive studies have implemented reinforcement learning (RL) in RMS for dynamic scheduling tasks, with Q-learning algorithms, as seen in Aydin and Öztemel [7] and Wang and Usher [8], demonstrating RL agents to outperform heuristic methods. Additionally, RL has been employed for higher-level reconfiguration decisions in RMS, with McDonnell et al. [9] optimizing machine reconfiguration and Luo [10] utilizing deep RL to solve complex scheduling problems. Advanced reinforcement learning (RL) algorithms, such as A2C, demonstrate potential in optimizing scheduling and reconfiguration in collaborative real-time energy management systems (RMS), as demonstrated by Yang and Xu [11]. Nonetheless, challenges like managing extensive action spaces necessitate further research. RL, including deep RL, represents a critical tool for intelligent decision-making in next-generation RMS that prioritises adaptability.

For modelling, simulation, and optimization of RMS, digital twins (DTs) are gaining traction. Huang et al. [12] and Liu et al. [13] proposed DT frameworks enhancing smart manufacturing systems and RMS reconfigurability. Tang et al. [14] and Zhong et al. [15] showcased DT frameworks and algorithms for decision support and optimal reconfiguration in RMS. Tao et al. [16] and Guo et al. [17] extended DT applications to real-time monitoring and predictive maintenance in RMS. Despite progress, challenges such as managing uncertainty of DTs and simplifying models continue. Exciting future research avenues include integrating DTs with AI and using them for quick RMS reconfiguration. DT technology offers tremendous potential for achieving the flexibility and responsiveness required in next-generation RMS.

III. Background

Reinforcement learning (RL) has emerged as a subfield of machine learning that is particularly

well-suited to applications such as robot control and adaptive manufacturing systems, where autonomous online learning is critical. Unlike supervised learning methods that rely on labelled training data, RL equips software agents with the ability to learn optimal actions through trial-and-error interactions in dynamic environments. This distinction makes RL a versatile framework for creating self-learning agents. At its core, RL is based on Markov Decision Processes (MDPs), which provide the mathematical models that include key components such as states, actions, transitions, and rewards. Finding the optimal policy that maximises the cumulative reward is the central challenge in RL. To enable the decomposition of complex MDPs into more tractable subproblems, Bellman's equation provides a recursive relationship between value functions that facilitates iterative policy refinement. A seminal RL algorithm called Q-learning, introduced by Watkins in 1989, incrementally updates action values or "Q" based on the Bellman equation to incrementally improve policies. To handle high-dimensional state spaces, Deep Q-Learning (DQL) combines deep neural networks with Q-learning. Crucially, innovations such as experience replay, introduced by Lin in 1993, and extensions such as Double DQN and Dueling DQN have improved the effectiveness of DQL in complex environments. Together, these advances have firmly established RL as a versatile framework for creating self-learning agents.

IV. Deep Reinforcement Learning - Digital Twin Contribution

A. Reinforcement Learning Environment Modelling

This section outlines the hierarchical deep reinforcement learning (DRL) control developed for integrated management within the RMS. The top-level RMS environment comprises several Reconfigurable Machine Tool (RMT) sub-environments, each containing two agents: a scheduling agent (SCHED) and a reconfiguration agent (RECONF) inspired from the work of yang and Xu [**?**].

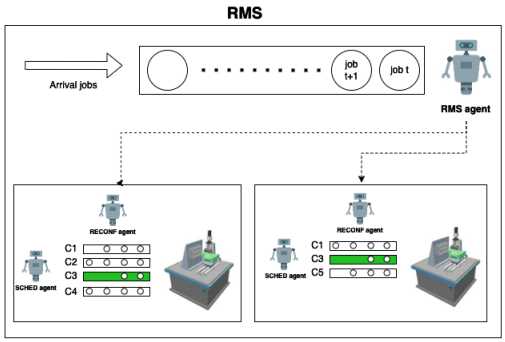

Fig. 1. System environments and agents Modelling

Input: Initial State of RMS and RMTs, Job Queue
Output: Optimized Scheduling and Reconfiguration Decisions
Initialize: RMS, RECONF, and SCHED agents;
while *Job Queue is not empty* **do**
  Digital Twin Data Acquisition:;
    Collect real-time data from Digital Twins of RMTs;
  RMS Hierarchical Environment:;
    Select an RMT to assign the next pending job based on aggregated state data;
    Step the RMTs to simulate the manufacturing system;
  Reward Function for RMS:;
    Calculate the average reward from each RMT environment;
  Actions for RMS:;
    Select one of the 5 actions to assign each pending job to an RMT, considering Digital Twin insights;
  State Features for RMS:;
    Update state features with information from Digital Twins;
  RMT Multi-agent Environment:;
    RECONF and SCHED agents make decisions, incorporating Digital Twin recommendations;
  Reconfiguration Agent (RECONF):;
    Calculate reward function;
    Select one of the 4 actions, considering Digital Twin insights;
    Update state features;
  Scheduling Agent (SCHED):;
    Calculate reward function;
    Select one of the 5 actions, considering Digital Twin insights;
    Update state features;
end

Algorithm 1: DQL Solution for RMS with Digital Twin Integration

This hierarchical structure simplifies the control problem by segregating scheduling and reconfiguration tasks. It aligns with RMS objectives of modularity and adaptability, facilitating the integration of additional tasks like transportation or maintenance.

B. Reconfiguration Agent (RECONF)

1) Reward function: RECONF manages dynamic reconfiguration of manufacturing system

modules. Its action space comprises four distinct actions (act1-act4) aimed at selecting the optimal machine configuration based on various operational conditions.

$$R = -\frac{1}{t_{S'} - t_S} \left(TP_{BF} + TP_{FNS} \right) \tag{1}$$

$$\mathrm{T\,P}_{BF} = \sum_{k=1}^{X} \sum_{j=1}^{n_k} \alpha_j z_{j_{S'}} \left[t_{S'} - \max\left(t_S, d_j \right) \right]$$

$$\mathrm{z}_{j_{S'}} = \begin{cases} 1, & \text{if } d_j < t_{S'} \\ 0, & \text{else} \end{cases}$$

$$\mathrm{T\ P}_{FNS} = \sum_{j=1}^{n_{FNS}} \alpha_j z_{jC} \left[C_j - \max\left(t_S, d_j \right) \right]$$

$$\mathrm{z}_{jC} = \begin{cases} 1, & \text{if } d_j < C_j \\ 0, & \text{else} \end{cases}$$

where TP_{BF}, TP_{FNS} are the newly added tardiness costs of jobs in queue buffers and finished jobs i respectively in the reconfiguration interval $(t_S, t_{S'})$ and α_j, C_j, d_j represent job unit tardiness cost, completion time after scheduling, due date.

2) Reconfiguration Trigger: To reduce reconfiguration frequency, reconfiguration is triggered only when:

- The current buffer is empty, or
- Overdue jobs exist and the current buffer has relatively low average tardiness, or
- The number of finished jobs from the current buffer exceeds a threshold and it has relatively low average tardiness.

This heuristic aims to balance reconfiguration and scheduling optimization. RECONF learns when reconfiguration is beneficial based on the manufacturing state.

3) Actions: it is imperative for Reconfigurable Control Framework (RECONF) to adapt its actions based on the specific machine production status it encounters. To facilitate this adaptability, RECONF's action space must encompass a range of candidate actions tailored to various operational conditions. Furthermore, the design of this action space should prioritize efficiency and simplicity. RECONF achieves this by defining four distinct actions, denoted as act1 through act4. Each of these actions corresponds to a unique strategy for selecting a candidate machine configuration (k) from the set 1, 2, ..., X. Each strategy focuses solely on jobs within the corresponding best-fit (BFk) production mode. In situations where multiple production modes exhibit equal merit within a strategy, RECONF opts for the first-best configuration available. Importantly, if the newly selected configuration aligns with the current configuration (k'), no machine reconfiguration is initiated. These four RECONF actions (act1-act4) are pivotal in ensuring adaptive and efficient control within dynamic manufacturing environments.

The four RECONF actions $(act_1 - act_4)$ are as follows:

(1) act_1: Select the machine configuration k that has the largest total ψ_j, where ψ_j denotes the current tardiness cost of job j:

$$act_1 =_k \left(\sum_{j=1}^{n_k} \psi_j \right), \quad k = 1, 2, ..., X \tag{2}$$

Where ψ_j is calculated by:

$$\psi_j = \begin{cases} \alpha_j, & \text{if } d_j < t_c \\ 0, & \text{else} \end{cases}$$

This action aims to switch to the configuration currently accruing the maximum total tardiness cost per second, in order to prioritize reducing delays for those jobs.

(2) A_2: Select the machine configuration k with the largest average ψ_j:

$$act_2 =_k \left(\frac{1}{n_k} \sum_{j=1}^{n_k} \psi_j \right), \quad k = 1, 2, ..., X \tag{3}$$

act_2 has advantages when a mode does not have the maximum total ψ_j due to few jobs, but has a high average.

(3) act_3: Select the machine configuration with the largest number of jobs:

$$act_3 =_k (n_k), \quad k = 1, 2, ..., X \tag{4}$$

Useful when no jobs are overdue or the number of waiting jobs dominates the decision.

(4) act_4: Select the machine configuration with minimum average safe time ST_j:

$$act_4 =_k \left(\frac{1}{n_k} \sum_{j=1}^{n_k} ST_j \right), \quad k = 1, 2, ..., X \quad (5)$$

Where:

$$ST_j = d_j - \sum_{i=1}^{m} t_{ij} - t_c \quad (6)$$

The four actions provide expressive reconfiguration capabilities based on assessments of tardiness and waiting time across production modes.

4) State Features and Observation Space: A DRL agent's choice of action depends on its assessment of the current state. It's essential that state features contain all relevant information needed for action evaluation. To ensure accuracy and minimize extraneous information, these state features should be streamlined and closely linked to the actionable data. In line with this principle, the Reconfigurable Control Framework has introduced four distinct state features, denoted as Feat1, Featt2, ..., Featt4, which are directly tailored to the specifics of the action space. These state features serve as essential components in shaping the agent's decision process.

Four state features $(Feat_1, Feat_2, \ldots, Feat_4)$, which are directly related to the action spaces, are designed for RECONF. These features are as follows ($\forall k = 1, 2, \ldots, X$:):

1 : The number of jobs in each BF_k,
$Feat_1 = \{n_1, n_2, \ldots, n_k, \ldots, n_X\}$.

2 : Total current unit tardiness cost of jobs in each BF_k

$$Feat_2 = \left\{ \sum_{j=1}^{n_1} \psi_j, \sum_{j=1}^{n_2} \psi_j, \ldots, \sum_{j=1}^{n_k} \psi_j, \ldots, \sum_{j=1}^{n_X} \psi_j \right\}.$$

3 :Average current unit tardiness cost of jobs in each BF_k

$$Feat_3 = \left\{ \frac{1}{n_1} \sum_{j=1}^{n_1} \psi_j, \frac{1}{n_2} \sum_{j=1}^{n_2} \psi_j, \ldots, \frac{1}{n_X} \sum_{j=1}^{n_X} \psi_j \right\}$$

4 :Average safe time of jobs in each BF_k.

$$Feat_4 = \left\{ \frac{1}{n_1} \sum_{j=1}^{n_1} ST_j, \frac{1}{n_2} \sum_{j=1}^{n_2} ST_j, \ldots \frac{1}{n_X} \sum_{j=1}^{nx} ST_j \right\}$$

The average duration of safe time for jobs within each best-fit production mode (BFk, where k ranges from 1 to X) is a pivotal consideration. It's worth noting that each state feature is structured as an array. To comprehensively capture the essence of each state feature, four statistical measures—namely, the maximum, minimum, average, and standard deviation—are meticulously computed, offering insights into the data's characteristics. As a result, the cumulative state space dimensions for Reconfigurable Control Framework (RCF) amount to 16, given that each of the four state features undergoes this comprehensive analysis. In order to normalize the state features effectively, the min-max normalization technique is employed.

C. Scheduling Agent (SCHED)

SCHED optimizes job scheduling within the current machine configuration. It possesses five actions to select the next job based on different selection strategies.

1) reward function: The increase in tardiness over a scheduling step [ts, ts'] comes from jobs waiting in the current buffer BFk' It is calculated as:

2) Actions: The SCHED agent has five actions for selecting the next job to schedule from the current buffer BFk'. Each action corresponds to a different job selection strategy based on factors like tardiness, due date, processing time, etc. By providing diverse scheduling options tied to assessments of the state, SCHED has expressive capability to learn nuanced scheduling policies.

The five SCHED actions are:

The act_1 selects a job with the max tardiness cost starting from current system time:
(1) $act_1 = \arg\max_j (\psi_j), \quad j \in BF_{k'}$

The act_2 selects a job with the max unit tardiness cost:
(2) $act_2 = \arg\max_j (\alpha_j), \quad j \in BF_{k'}$

The act_3 selects a job with the minimum safe time:
(3) $act_3 = \arg\min_j (ST_j), \quad j \in BF_{k'}$

The act_4 selects a job with the nearest due date:

(4) $act_4 = \arg\min_j (d_j), \quad j \in BF_{k'}$

The act_5 selects a job with the minimum processing time:

(5) $act_5 = \arg\min_j (\sum_{i=1}^{m} t_{ij}), \quad j \in BF_{k'}$

The diverse set of actions aim to provide SCHED with flexibility in learning appropriate scheduling policies across scenarios.

3) State Features and Observation Space: The state features and observation space for the SCHED agent aim to provide relevant information about the current staus of machine to inform its scheduling decisions. Carefully representing the state enables SCHED to learn policies that map conditions to appropriate job sequencing actions.

(1) $feat_1 = n_{k'}$. The number of jobs in $\mathrm{BF}_{k'}$.

(2) $feat_2 = \{\psi_j\}, j = 1, 2, \ldots, n_{k'}$. Current unit tardiness cost of jobs in BF_k.

(3) $\mathrm{feat}_3 = \{\alpha_j\}, j = 1, 2, \ldots, n_{k'}$. Unit tardiness cost α_j of jobs in BF_k.

(4) $feat_4 = \{ST_j\}, j = 1, 2, \ldots, n_{k'}$. Safe time ST_j of jobs in BF_k.

(5) $feat_5 = \{d_j\}, j = 1, 2, \ldots, n_{k'}$. Due date d_j of jobs in BF_k.

(7) $feat_7 = \{\sum_{i=1}^{m} t_{ij},\}, j = 1, 2, \ldots, n_{k'}$. Total processing time of jobs in BF_k.

Several of the defined state features for SCHED are arrays, such as the unit tardiness costs and safe times of the waiting jobs. To extract useful information from these arrays, statistical characteristics are calculated, including the maximum, minimum, average, and standard deviation. With 5 array features, this results in 5 * 4 = 20 state variables. Additionally, for the unit tardiness costs array, the number of zero values and non-zero values are included as extra state features. In total then, the SCHED state representation consists of $1+5*4+2 = 23$ dimensions. Like with RECONF, min-max normalization is employed to scale the state features to a standardized range between 0 and 1. This normalization prevents any single feature from dominating. The 23-dimensional observation aims to provide SCHED with sufficient relevant environment information to learn effective scheduling policies.

$$R = -\frac{1}{t_{s'} - t_s}(TP_{BF_{k'}}) \tag{7}$$

$$\mathrm{TP}_{BF_{k'}} = \sum_{j=1}^{n_{k'}} z_{js'} \alpha_j [t_{s'} - \max(t_s, d_j)]$$

$$\mathrm{z}_{j_{s'}} = \begin{cases} 1, & d_j < t_{s'} \\ 0, & \text{else} \end{cases}$$

where $z_{js'}$ indicates whether job j is overdue at $t_{s'}$.

By rewarding low incremental tardiness from scheduling actions, SCD can learn when to schedule which job to optimize sequencing.

D. RMS Hierarchical environment

The RMS environment coordinates the RMT sub-environments, each modeling a machine with a RECONF and SCHED agent. The decentralized agent structure facilitates independent learning of reconfiguration and scheduling policies.

E. Reward Function

The RMS reward is the average of the rewards obtained by each RMT environment at each timestep. This aligns the global RMS objective of maximizing manufacturing productivity with the machine-level RMT rewards designed to minimize job tardiness costs.

F. Actions

The RMS actions correspond to assigning each pending job to one of the RMT environments. The target RMT is selected based on aggregated state measures to optimize system-level performance.

Specifically, there are 5 actions for the RMS agent to choose from:

Select the RMT with the job's configuration:

1) and minimum job's configuration buffer length
2) and maximum job's configuration buffer safe job time
3) and minimum job's configuration buffer total tardiness
4) and minimum job's configuration buffer average tardiness
5) and already running the job's configuration, or the one with minimum configuration buffer length

By dispatching jobs to machines based on factors like queue occupancy, safe time, and tardiness costs, the RMS can coordinate assignments to improve system-level scheduling performance.

G. State Features

The RMS state representation summarizes high-level state information across the underlying RMT environments, including normalized measures of:

Job queue lengths per RMT Total tardiness costs per RMT Average tardiness per RMT Job safe times per RMT

It tracks RMT-level features like queue occupancy and scheduling metrics to inform system-level job assignment decisions. Additionally, it encodes current job configurations and selected RMT configurations. This compact state representation allows the RMS to assess global shop status for effective joint reconfiguration and scheduling.

Input: Environment Configuration *env_config*
Output: Trained policies and results

Initialization: Initialize environment with *env_config*;
Initialize DDQN networks and replay buffer;
Initialize agent models and target networks;
Initialize Q-networks with weights θ and θ_{target};
Initialize target network update frequency;
Initialize exploration strategy parameters;
for *episode = 1 to num_episodes* **do**
  Reset environment with `reset` function;
  Initialize state s;
  Initialize total episode reward $R \leftarrow 0$;
  while *episode not finished* **do**
    for *agent in agents* **do**
      `// Get action from agent using DDQN`
      $act[agent] \leftarrow$ `agent.get_ddqn_action(`s`);`
    end
    `// Execute actions in environment`
    $obs, rew, terminateds, truncateds, info \leftarrow$ `step(`act`);`
    for *agent in agents* **do**
      `// Update agent's replay buffer`
      `agent.update_replay_buffer(`$obs[agent], act[agent], rew[agent], terminateds[agent]$`);`
      `// Sample a random minibatch from the replay buffer` D
      $batch \leftarrow$ `replay_buffer.sample_batch();`
      `// Calculate target Q-values using the target network:` $Q_{\text{target}}(s', a')$
      $Q_{\text{target}} \leftarrow$ `target_network(`$batch.next\_states$`);`
      `// Update Q-network using the DDQN loss`
      $\mathcal{L}(\theta) \leftarrow \frac{1}{N} \sum_i \left(Q(s,a) - (r + \gamma Q_{\text{target}}(s', a'))\right)^2$;
      `// Update the target network weights` θ_{target}
      $\theta_{\text{target}} \leftarrow \tau\theta + (1-\tau)\theta_{\text{target}}$;
      `// Update state and episode reward`
      $s \leftarrow s'$;
      $R \leftarrow R + r$;
    end
    `// Update exploration strategy parameters`
    `update_exploration_strategy();`
  end
end

Algorithm 2: RMT Multi-Agent Environment training with DDQN

H. Digital Twin Architecture

To reflect the modularity and adaptability of the physical Reconfigurable Manufacturing System (RMS), a digital twin architecture is developed where each Reconfigurable Machine Tool (RMT) is represented as an individual digital twin. This approach allows for a decentralized and modular representation in line with the objectives of the RMS.

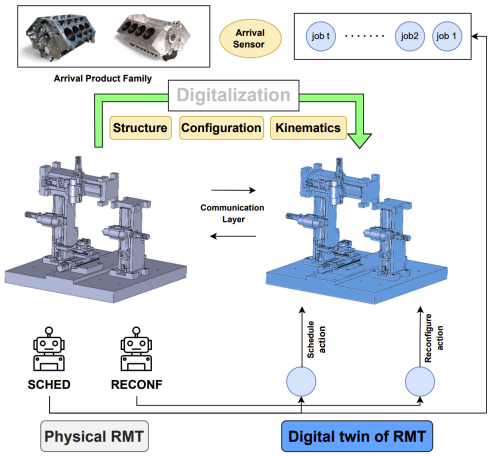

Fig. 2. Illustration of RMT Digital Twin Architecture

The digitisation process of each RMT goes beyond the replication of its external appearance. It involves the complete digitisation of three crucial aspects:

1) **Digitalization of Structure:** This involves creating a digital replica of the basic modules and components of the RMT. This includes modelling the physical structure and framework that make up the RMT. The digital twin captures the physical dimensions, materials and geometric configurations of these components, ensuring an accurate representation.
2) **Digitalization of Configuration:** Another key facet is the digitisation of the RMT configuration. This involves modelling the availability and arrangement of tools and fixtures on the RMT.
3) **Digitalization of Kinematics:** Kinematics refers to the study of motion and mechanical behaviour within the RMT. Kinematics digitisation involves the creation of mathematical models that describe the movements, motions and interactions of the components of the RMT, including the Tool Approach Directions (TADs) of the RMT.

By digitally capturing these aspects of each RMT, the digital twin architecture enables a holistic representation of the RMT's behaviour, both structurally and functionally. This comprehensive approach not only facilitates real-time monitoring and simulation, but also facilitates decision-making within the RMS, in line with the system's goals of adaptability and modularity.

I. Network and Communication Protocol

To enable seamless and interoperable data exchange between the physical Reconfigurable Machine Tools (RMTs) and their digital twins, the Open Platform Communications Unified Architecture (OPC UA) standard is adopted. OPC UA provides a platform-independent service-oriented architecture with robust information modeling and security capabilities suited for industrial control systems. Within the Digital Twin framework, OPC UA facilitates bidirectional communication between the physical RMTs and virtual models using a client-server architecture like illustrated in figure 3.

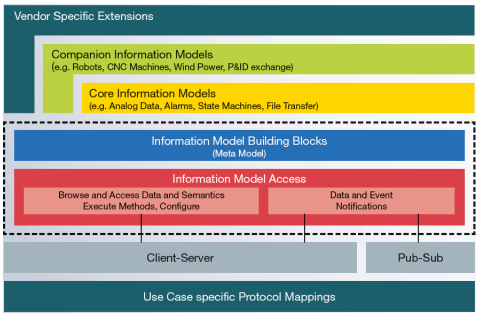

Fig. 3. Illustration of OPC UA Architecture [18]

The OPC UA server resides in the physical RMT's control system, making data such sensor measurements and control signals available through standard interfaces. This data is consumed by the Digital Twin OPC UA client to update the virtual model. Conversely, commands from the Digital Twin OPC UA server are transmitted to orchestrate the behavior of the physical RMT OPC UA client. By leveraging OPC UA, the Digital Twin Architecture enables seamless integration and interoperation between diverse industrial assets and management systems. Its platform-neutral design and focus on interoperability make it an ideal communication middleware for realizing the vision of adaptable and intelligent Reconfigurable Manufacturing Systems.

J. Predictive Maintenance

Predictive maintenance is a critical aspect of the Digital Twin's role in the Reconfigurable Manufacturing System (RMS). By using real-time data and advanced analytics, the Digital Twin enables proactive maintenance strategies that can significantly improve the reliability and uptime of the RMS. This subsection outlines how predictive maintenance is integrated into the Digital Twin architecture:

1) **Condition Monitoring:** The Digital Twin continuously collects data from various sensors placed within the RMS, including those within each Reconfigurable Machine Tool (RMT). These sensors monitor critical parameters such as job completion status and reconfiguration status. The data from these sensors are used to assess the current condition of the RMTs and the overall RMS.
2) **Maintenance Alerts and Recommendations:** When the Digital Twin identifies potential maintenance requirements, it generates maintenance alerts. These alerts are communicated to the RMS operators or maintenance personnel in real-time. The alerts provide detailed information about the nature of the issue, its severity, and recommended actions for maintenance or repairs.
3) **Historical Maintenance Data:** Data on past maintenance events and their outcomes are stored and analyzed within the Digital Twin. This historical data allows for continuous improvement of maintenance strategies and predictive models. Lessons learned from past maintenance incidents contribute to more accurate predictions and optimized maintenance procedures over time.

Predictive maintenance, facilitated by the Digital Twin, significantly enhances the overall reliability and availability of the Reconfigurable Manufacturing System. It reduces unplanned downtime, lowers maintenance costs, and ensures that the RMS operates at peak efficiency by addressing maintenance needs proactively.

V. Conclusion and Future Work

In summary, our Modular Deep Reinforcement Learning-Digital Twin Framework represents a game-changing paradigm for the control and optimisation of reconfigurable manufacturing systems (RMS). This framework embodies the convergence

of two transformative technologies: Deep Reinforcement Learning (DRL) and Digital Twin technology, all orchestrated within a modular architecture. This combination produces a rich set of capabilities that enable us to move towards full control of RMS operations and significantly increase their resilience in the face of dynamic manufacturing environments.

The power of this combination lies in the complementary strengths of each component. Digital Twins, with their ability to replicate the structure, configuration and kinematics of reconfigurable machine tools (RMTs), provide a holistic and real-time representation of the RMS. This representation allows us to predict maintenance needs, simulate different operating scenarios and optimise job scheduling and reconfiguration decisions with unparalleled accuracy. In essence, Digital Twins turn the physical RMS into a digital playground where we can proactively anticipate, adapt and optimise operations.

On the other hand, Deep Reinforcement Learning agents nested within the hierarchical structure of the RMS give the system adaptability and intelligence. These agents continuously learn from their interactions with the digital twins and the physical RMS, and adjust their decisions in real time. This adaptability ensures that the RMS remains responsive to changing conditions, aligns its operations with production requirements, and handles unforeseen challenges with grace.

In addition, our framework goes beyond operational efficiency by laying the groundwork for full control of the RMS. The Digital Twins provide a digital bridge between the physical and virtual worlds, enabling operators and decision-makers to visualise and fine-tune RMS behaviour with unprecedented granularity. By combining the insights of Digital Twins with the adaptability of DRL agents, we create a dynamic and intelligent RMS control system capable of responding precisely to changing production needs.

In the context of resilience, our framework acts as a bastion against unplanned downtime and production disruptions. The predictive maintenance capabilities derived from Digital Twins ensure that maintenance tasks are carried out proactively, reducing costly interruptions. Meanwhile, the adaptability of DRL agents ensures that the RMS thrives in unpredictable and dynamic manufacturing environments, ensuring consistent productivity.

REFERENCES

[1] R. Rajkumar, G. Ravi, and A. Zalzala, "Recent advances in evolutionary and adaptable manufacturing systems," *International Journal of Production Research*, vol. 48, no. 22, pp. 6675–6696, 2010.

[2] Y. Koren, U. Heisel, F. Jovane *et al.*, "Reconfigurable manufacturing systems," *CIRP Annals*, vol. 48, no. 2, pp. 527–540, 1999.

[3] A. Bilberg, R. Malik, and K. Bøgh, "New model for development and manufacturing of tailored solutions in the industrial market," *Journal of Manufacturing Systems*, vol. 31, no. 3, pp. 367–374, 2012.

[4] Z. Li, L. Li, and A. Bilberg, "Design and implementation of a reconfigurable manufacturing system," *International Journal of Advanced Manufacturing Technology*, vol. 39, pp. 1181–1191, 2008.

[5] A. Dashchenko, *Reconfigurable manufacturing systems and transformable factories*. Springer Science & Business Media, 2006.

[6] S. Riedmiller and M. Riedmiller, "A neural reinforcement learning approach to learn local dispatching policies in production scheduling," 07 2000.

[7] M. Aydin and E. Oztemel, "Dynamic job-shop scheduling using reinforcement learning agents," *ROBOTICS AND AUTONOMOUS SYSTEMS*, vol. 33, 11 2000.

[8] Y.-C. Wang and J. Usher, "Learning policies for single machine job dispatching," *Robotics and Computer-Integrated Manufacturing*, vol. 20, pp. 553–562, 12 2004.

[9] P. McDonnell, S. Joshi, and R. Qiu, "A learning approach to enhancing machine reconfiguration decision-making games in a hierarchical manufacturing environment," *International Journal of Production Research*, vol. 43, pp. 4321–4334, 10 2005.

[10] S. Luo, "Dynamic scheduling for flexible job shop with new job insertions by deep reinforcement learning," *Applied Soft Computing*, vol. 91, p. 106208, 03 2020.

[11] S. Yang and Z. Xu, "Intelligent scheduling and reconfiguration via deep reinforcement learning in smart manufacturing," *International Journal of Production Research*, pp. 1–18, 2021.

[12] S. Huang, G. Wang, and Y. Yan, "Building blocks for digital twin of reconfigurable machine tools from design perspective," *International Journal of Production Research*, vol. 60, pp. 1–15, 12 2020.

[13] Z. Liu, T. Wang, Y. Zhou, W. Zhao, M. Zheng, Z. Ke, and X. Zhao, "Digital twin-based reconfiguration time point prediction method for reconfigurable manufacturing systems," *Journal of Physics: Conference Series*, vol. 2173, no. 1, p. 012058, jan 2022. [Online]. Available: https://dx.doi.org/10.1088/1742-6596/2173/1/012058

[14] J. Tang, C. Emmanouilidis, and K. Salonitis, "Reconfigurable manufacturing systems characteristics in digital twin context," *IFAC-PapersOnLine*, vol. 53, pp. 10 585–10 590, 01 2020.

[15] J. Kombaya Touckia, N. Hamani, and L. Kermad, "Digital twin framework for reconfigurable manufacturing systems (rmss): design and simulation," *The International Journal of Advanced Manufacturing Technology*, vol. 120, 06 2022.

[16] Y. Cai, Y. Wang, and M. Burnett, "Using augmented reality to build digital twin for reconfigurable additive manufacturing system," *Journal of Manufacturing Systems*, vol. 56, 05 2020.

[17] S. Martinez, A. Mariño, S. Sanchez, A. Montes, J. Triana, G. Barbieri, S. Abolghasem, J. Vera, and M. Guevara, "A digital twin demonstrator to enable flexible manufacturing with robotics: a process supervision case study," *Production Manufacturing Research*, vol. 9, pp. 140–156, 08 2021.

[18] OPC. (2008) Unified architecture. [Online]. Available: https://opcfoundation.org/about/opc-technologies/opc-ua/