2026 ASEE ANNUAL CONFERENCE June 21–24, 2026 | Charlotte Convention Center, NC
Host Sponsor UNC Charlotte
Paper ID #52433

Bridging Reinforcement Learning Theory and Practice in Systems Engineering Education Through Simulation Environment Design

Srijana Raut, University of North Carolina at Charlotte
Dr. Lingxiang Yun, University of North Carolina at Charlotte

Dr. Lingxiang (Albert) Yun is an Assistant Professor in the Department of Industrial and Systems Engineering at UNC Charlotte. His research leverages artificial intelligence and machine learning to support data-driven decision-making in complex engineering systems. His areas of interest include sustainable and advanced manufacturing systems, energy system management, intelligent supply chain planning, and other related engineering applications.

Dr. Simon M. Hsiang P.E., North Carolina State University at Raleigh

Bridging Reinforcement Learning Theory and Practice in Systems Engineering Education Through Simulation Environment Design

Abstract

Reinforcement learning (RL) has emerged as a powerful tool for solving complex decision-making problems in engineering systems, from production scheduling to energy management. However, RL is often introduced to students through OpenAI Gym, the most widely used open-source RL training and testing environment, which primarily features game-based tasks such as CartPole, MountainCar, and LunarLander. While these environments are effective for teaching algorithm mechanics, they lack the context of real-world industrial systems, making it difficult for students, especially those in systems engineering (SE), to recognize the practical relevance of RL techniques.

This work presents the development of an open-source simulation environment designed to support RL education using SE context. The environment includes system models such as serial and assembly lines, and it supports the exploration of problems like resource allocation and dynamic scheduling under uncertainty. The simulation environment is designed using Python to seamlessly integrate with existing RL toolkits. The environment provides the necessary interfaces and data structures to support both student-coded RL algorithms and general-purpose libraries such as Stable-Baselines3 (SB3), enabling flexible demonstration and experimentation in instructional settings. The instructional approach encourages students to engage with key RL concepts in a realistic SE setting, such as trying new, uncertain options to find potential future benefits (exploration) versus using existing, known-to-be-good options to maximize immediate rewards (exploitation). The environment is designed to be integrated as a project-based learning component to help students better understand how RL can support decision-making in complex, uncertain systems. This paper focuses on the design and instructional integration of the environment; classroom deployment and empirical assessment are planned as future work.

Keywords: Reinforcement learning, Project-based learning, Open-source teaching tools

1 Introduction

Reinforcement learning (RL) has emerged as an important decision-making tool for engineering systems that operate under uncertainty and evolve over time. By modeling control and scheduling problems as iterative interactions between an agent and a dynamic environment, RL provides a flexible framework for learning policies that respond to short-term disturbances while accounting for long-term system performance. As a result, RL has been increasingly explored in engineering applications such as manufacturing operations [1], [2], energy management [3], [4], and other complex systems where decisions must be made repeatedly in the presence of stochastic system dynamics. These characteristics closely align with the core concerns of systems engineering (SE), which emphasize the modeling, analysis, and management of interconnected components and their interactions over time.

In both academic courses and widely used textbooks, RL is commonly introduced through standardized benchmark environments that emphasize algorithmic mechanics rather than system context. Foundational RL textbooks, such as Reinforcement Learning: An Introduction [5] and

Deep Reinforcement Learning in Action [6], frequently illustrate key concepts using toy examples that are easy to formalize and analyze. Similarly, many RL courses rely on benchmark environments provided by OpenAI Gym, which has become a standard for RL instruction. These environments typically feature game-like tasks, such as CartPole, MountainCar, and LunarLander, that offer well-defined state and action spaces and fast simulation cycles, making them convenient for demonstrating core algorithms and training procedures [7].

While benchmark-based environments are effective for illustrating the mechanics of RL concepts and algorithms, they provide limited support for helping SE students connect RL concepts to real engineering systems. In practical engineering applications, the formulation of the decision-making problem, including how the system is defined in terms of states, actions, and rewards, is often as important as, if not more important than, the choice of a specific RL algorithm [8]. Many RL studies demonstrate that system representation strongly influences learning performance, policy behavior, and implementation robustness [8], [9]. However, these game-like benchmark problems rely on simplified and fully specified system definitions, allowing students to focus primarily on algorithm implementation rather than on the modeling decisions that characterize real-world systems. As a result, while such environments are effective for understanding how RL algorithms operate under well-defined settings, they provide limited opportunity for students to practice applying RL in complex systems where decision makers must select observable states, account for feasible action constraints, and carefully design reward structures to reflect system-level objectives and trade-offs.

To address this gap, this paper presents an RL-enabled manufacturing system simulation environment designed to support RL education in a SE context. The environment models configurable manufacturing systems and includes a graphical user interface (GUI) that allows users to manually modify system configurations. Under each configuration, the environment provides a rich set of observable system states, including stochastic processing times at individual machines, mean processing rates, work-in-process (WIP) levels at each buffer, and total system WIP, etc. These observable states may exceed what is strictly necessary for a specific control task, intentionally allowing students to explore how state selection influences learning behavior and policy performance. The environment also supports multiple control actions, such as constant work-in-progress (CONWIP) control or simple machine up/down decisions, along with alternative reward definitions that reflect different system-level objectives. By interacting with this flexible system formulation, students can examine fundamental RL concepts in an engineering setting [10], e.g., the interpretation of Markov decision processes, the differences between dynamic programming, Monte Carlo methods, and temporal-difference learning, and the conditions under which each approach is appropriate. Moreover, the large state space, diverse action options, and multiple reward formulations provide opportunities for students to observe how different modeling choices for the same physical system can significantly affect RL training behavior and performance [11], [12].

The contributions of this paper are as follows. First, it presents an RL-enabled manufacturing system simulation environment designed to bridge RL concepts with SE applications. Second, it provides an educational framing that highlights how system modeling choices, such as state representation, action selection, and reward design, affect RL behavior in complex engineering systems. Third, the paper discusses how the proposed environment can be integrated into a RL curriculum as a project-based instructional component. The present work focuses on

environment development and curricular alignment; classroom implementation and assessment are reserved for future study.

The rest of the paper is organized as follows. Section 2 outlines the educational motivation and context for using manufacturing systems in RL education. Section 3 describes the simulation environment, the RL formulation, and illustrative training behavior. Section 4 discusses instructional integration and course alignment, and Section 5 concludes with a discussion of future classroom deployment.

2 Educational Motivation for RL in Systems Engineering Context

2.1 RL for System Decision-Making

From a systems engineering perspective, RL can be viewed as a framework for modeling and solving sequential decision-making problems in engineering systems. An RL problem formulation requires defining the system state, control actions, transition dynamics, and performance objectives, which together describe how decisions influence system behavior over time. These elements closely mirror the core components of SE models, where system performance emerges from interactions among subsystems, resource constraints, and stochastic processes [13]. As a result, applying RL in engineering contexts is not solely an algorithmic exercise, but also a modeling task that requires careful abstraction of system structure and decision logic.

In practical applications, different representations of the same physical system can lead to substantially different learning dynamics and control outcomes. Choices regarding which variables should be observed, how actions are constrained, and how rewards are defined directly affect the decision-making approach. For example, fully specified models may support planning-based methods such as dynamic programming, whereas complex system without explicate state transition models necessitate model-free learning techniques. Also, the choice between on-policy and off-policy learning is closely related to system features such as the cost of exploration, the ability to collect data under fixed baseline policies, and the stability requirements of system operation, which in turn affects data efficiency and learning stability [8]. These considerations illustrate that effective application of RL in engineering contexts depends on aligning learning strategies with system characteristics, such as uncertainty, observability, and operational constraints, rather than relying on algorithm selection alone.

2.2 Educational Implications for Systems Engineering Students

Viewing RL through a systems engineering lens has important implications for how RL concepts are introduced to SE students [14]. In this context, learning RL extends beyond understanding algorithmic update rules to developing the ability to reason about how modeling choices shape decision-making behavior and system performance. When students engage with RL in engineering systems, they are required to reason about how modeling assumptions translate into decision-making behavior [11]. These considerations are central to systems engineering practice and naturally align RL with broader system analysis and design principles [15], [16].

The proposed RL-enabled manufacturing system simulation environment supports this perspective by providing a system context in which both fundamental RL concepts and engineering applications can be illustrated. For example, the environment enables illustration of

core RL concepts such as Markov decision processes, value estimation, policy evaluation, and exploration–exploitation trade-offs, as well as the practical differences among dynamic programming, model-free prediction and control, and policy-based methods. Additionally, students can examine how expanding or reducing the state representation affects learning efficiency and policy robustness, how conservative and aggressive action constraints influence system stability, and how different reward formulations emphasize competing objectives such as throughput, congestion, or WIP utilization. By grounding these concepts in a realistic system setting, the environment allows students to observe how algorithmic behavior and system performance co-evolve, reinforcing the idea that effective application of RL in engineering contexts requires thoughtful problem formulation in addition to algorithm understanding [12].

3 Simulation Based RL Learning Environment

In RL, an agent learns a decision-making policy through repeated interaction with an environment by observing system states, selecting actions, and receiving feedback in the form of performance reward signals. Through this iterative process, the agent gradually improves its behavior based on accumulated experience rather than explicit system models. Figure 1 illustrates this interaction loop between the learning agent and the manufacturing system simulation environment, which forms the basis for the simulation and training framework described in the remainder of this section.

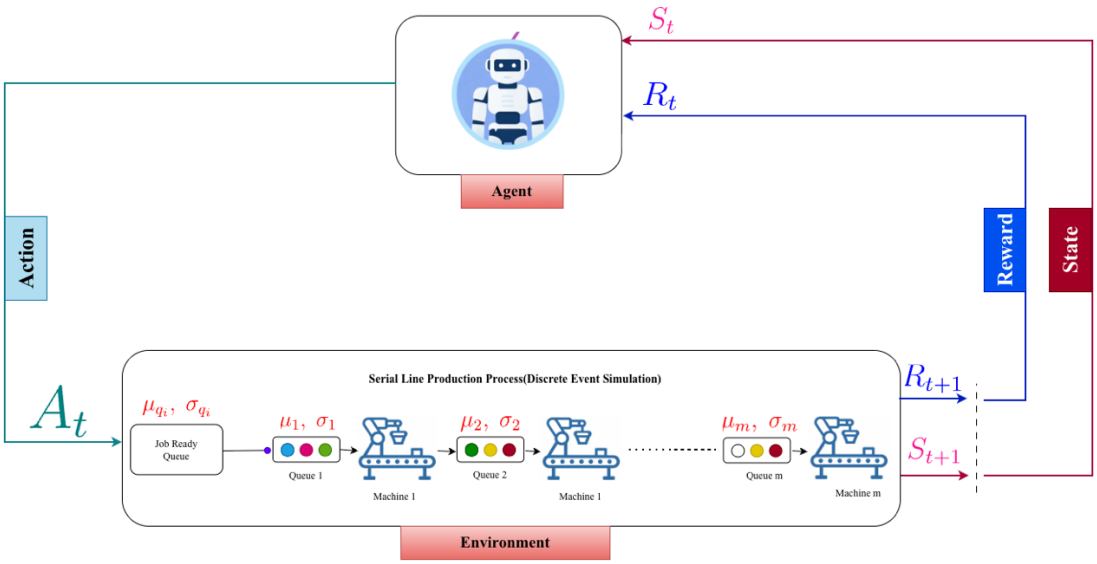

Figure 1: RL Interaction Loop Integrated with a Manufacturing Simulation System

3.1 Manufacturing System Simulation

The learning environment is based on a serial production line, a common system in manufacturing. The system consists of multiple machines arranged sequentially. Each machine processes one job at a time, and jobs may wait in buffers between machines when downstream machines are busy. The system is implemented using discrete-event simulation (DES), which advances time only when key events occur, such as job arrivals, machine completions, or job transfers between buffers. This modeling approach closely reflects how real manufacturing systems operate.

To represent realistic operating conditions, processing times at each machine are stochastic. Each job's processing time is randomly sampled from a bounded distribution with a fixed mean and variability. This allows students to observe how randomness and variability propagate through the system and influence congestion, throughput, and cycle time.

Job releases into the system are controlled using a CONWIP policy. Under CONWIP, a new job is released only when another job completes, keeping the total number of jobs in the system approximately constant. This mechanism naturally exposes the fundamental trade-offs between WIP, Throughput (TH), and Cycle time (CT). Throughout the simulation, key system performance measures such as total WIP, machine queue lengths, and throughput are continuously recorded. These measures are used both for RL decision-making and for student interpretation and analysis. The simulation is implemented as a RL environment that follows the Gymnasium standard, which allows it to work easily with commonly used RL libraries in Python.

3.2 Reinforcement Learning Formulation

Reinforcement learning is used as a decision-making framework that interacts with the simulation over time. The three key components that can affect the RL performance are summarized in Table 1. How these components are presented in the proposed environment is further explained in the following sections.

Table 1: Core RL Components and Their Roles

RL Component	Description
State	Observable variables describing the current system condition (e.g., machine status, buffer levels)
Action	Control decisions applied to the system that influence its evolution
Reward	Scalar feedback evaluating system performance relative to operational objectives

3.2.1 State Space

The state represents what the agent can observe about the system at a given time. The environment intentionally provides a rich set of observable variables so students can explore how different state selections affect learning. Example of state variable include which collectively describe system congestion, variability and performances are summarized in Table 2.

Table 2: RL State Variables and Their Significance

State Variable	Mathematical Symbol	Why this variable is Useful
Mean processing time at machine i	$\mu_i(t)$	Reflects workload intensity at each machine

Variability processing time	$\sigma_i(t)$	Captures variability in processing behavior
Bottleneck utilization	$u_i(t)$	Indicates bottleneck utilization status
Mean queue length	$\mu_i^q(t)$	Reveals congestion accumulation patterns
Queue variance	$\sigma_i^q(t)$	Signals queue instability or fluctuation
Estimated cycle time	$CT(t) = \frac{WIP(t)}{TH(t)}$	Represents job completion duration
System throughput	$TH(t) = \frac{N_{comp}(\mathrm{t}) - N_{comp}(t - \Delta t)}{\Delta t}$	Reflects system production rate
Throughput trend	$\Delta TH(t) = TH(t) - TH(t-1)$	Indicates performance trends over time
Target throughput	TH_{Target}	Specifies the performance objective
Throughput difference	$\Delta TH(t)$	Quantifies deviation from the objective
Total system WIP	$WIP(t)$	Measures overall system congestion

3.2.2 Action Space

The simulation environment provides a range of possible control actions by RL agents. These actions can be freely chosen to change the system behavior, which summarized in Table 3.

Table 3: RL Action Variables

Action	**Mathematical Symbol**	**Effect on System**
CONWIP setpoint	$WIP_{setmax}(t)$	Regulates overall system congestion
Incremental WIP change	$\Delta WIP(t) = WIP_{set}(t) - WIP_{set}(t-1)$	Enables continuous control adjustments
On/off control	$u(t) \in \{0,1\}$	Control machine on or off

Release rate	$\lambda(t)$	Adjusts job entry rate into the system
Release interval	$\tau(t)$	Controls spacing between job releases

3.2.3 Reward Design

The reward provides feedback on how well the agent's action supports the system objective. The reward function can be designed based on need. Some examples of reward function can be used in the proposed framework are summarized in Table 4.

Table 4: Example of Reward Functions

Reward	**Mathematical Symbol**	**Control Objective**
Throughput tracking	$-\left(\Delta TH(t)\right)^2$	Pushes performance toward the target
Smooth tracking	$e^{-\left(\Delta TH(t)\right)^2}$	Provides gentle and stable feedback
WIP minimization	$-WIP(t)$	Encourages lean operation

3.3 Demonstrative Problem Setting and Results

This subsection demonstrates how RL operates in the proposed simulation environment. Rather than exposing all available variables, a compact and interpretable subset of states is used to clearly illustrate the learning process. In the demonstrative results, the objective is tracking a desired throughput level while maintaining stable operation. Thus, the selective states are.

1. Mean processing times (μ_i)

$$\mu_i = \frac{1}{J_i}\sum_{j=1}^{J_i} t_{i,j} \tag{1}$$

2. Standard deviation of processing times (σ_i)

$$\sigma_i = \sqrt{\frac{1}{J_i}\sum_{j=1}^{J_i} (\, t_{i,j} - \mu_i)^2} \tag{2}$$

where J_i denotes the number of jobs either waiting in the queue or currently being processed at machine *i.*

3. Mean queue length ($\mu_{q,1}$)

$$\mu_i^q = \frac{1}{T}\sum_{t=1}^{T} Q_i(t) \tag{3}$$

4. Standard deviation of queue ength ($\sigma_{q,i}$)

$$\sigma_i^q = \sqrt{\frac{1}{T}\sum_{t=1}^{T}\left(Q_i(t) - \mu_i^q\right)^2} \tag{4}$$

5. Throughput target $\left(TH_{Target}\right)$

The Throughput rate is measured on simulation clock, representing number of jobs per minute.

6. Throughput difference (ΔTH)

$$\Delta TH = \frac{TH_{Target} - TH_{Actual}}{TH_{Target}} \tag{5}$$

7. Total WIP (WIP_{total})

Total number of jobs currently in the production system.

The complete state vector formulation is follow the work in [17] and presented as follows.

$$S = \begin{bmatrix} \mu_1, \mu_2, \dots, \mu_m, \\ \sigma_1, \sigma_2, \dots, \sigma_m, \\ \mu_1^q, \mu_2^q, \dots, \mu_m^q, \\ \sigma_1^q, \sigma_2^q, \dots, \sigma_m^q, \\ TH_{Target}, \\ \Delta TH, \\ WIP_{total} \end{bmatrix} \tag{6}$$

At each decision step, the agent selects a discrete CONWIP setpoint, which defines the total allowable WIP in the system. Adjusting this setpoint indirectly controls job releases, thereby influencing congestion and throughput. The minimum WIP setpoint is 1, while the maximum WIP, denoted by w, is derived from the Practical Worst Case (PWC) throughput relationship. This relationship illustrates the diminishing returns of increasing WIP: beyond a critical threshold, additional WIP primarily leads to congestion rather than improved throughput. Specifically, the PWC throughput for a serial line with bottleneck rate r_b and critical WIP level W_0 is given by:

$$TH_{PWC} = \frac{w}{W_0 + w - 1/r_b} \tag{7}$$

Thus, the upper bound of CONWIP control action w can be expressed by:

$$w = \frac{TH_{Target}(W_0 - 1)}{r_b - TH_{Target}} \tag{8}$$

For example, a five-machine serial line with a bottleneck rate r_b of 0.1 jobs/min, a target throughput of 80% of capacity, and a critical WIP level $W_0 = 5$, the maximum allowable WIP is computed as $w = 16$.

Based on this calculation, the agent can choose a CONWIP level between 1 and 16. Each choice sets how many jobs are allowed in the system at one time. Because jobs must pass through several machines with random processing times, the effect of a decision is not immediate and is observed after some delay.

To guide learning toward the control objective, the reward is designed so that the agent is rewarded when the actual throughput is close to the target throughput and penalized as the deviation increases. To achieve this, a Gaussian-shaped reward function is employed:

$$r = \beta \cdot \left(e^{-\varepsilon^2/\sigma^2}\right) - 1 \tag{9}$$

where the reward depends on the normalized throughput error:

$$\varepsilon = \frac{|\ \Delta TH_r - \Delta TH\ |}{\Delta TH_r} \tag{10}$$

In this demonstration, the parameters are set to $\beta = 1$ and $\sigma = 0.2$. These values produce a smooth reward landscape that discourages aggressive or unstable control actions and promotes gradual convergence toward the desired throughput.

At each decision step, the agent selects a WIP level that determines how many jobs are allowed in the production system at one time. The reward is based on how close the actual system throughput is to the desired target. This design encourages the agent to adjust WIP smoothly and avoid unstable or overly aggressive control actions. During training, the throughput target can change, requiring the agent to adapt its decisions instead of learning a fixed control rule.

The learning agent is implemented using a Deep Q-Network (DQN). Layer normalization is applied to improve learning stability when system conditions change. Through repeated interaction with the environment, the agent learns which WIP levels perform best under different system states by trying actions, observing the resulting rewards, and learning from past experiences. This approach keeps the learning process easy to understand while remaining effective for controlling the system.

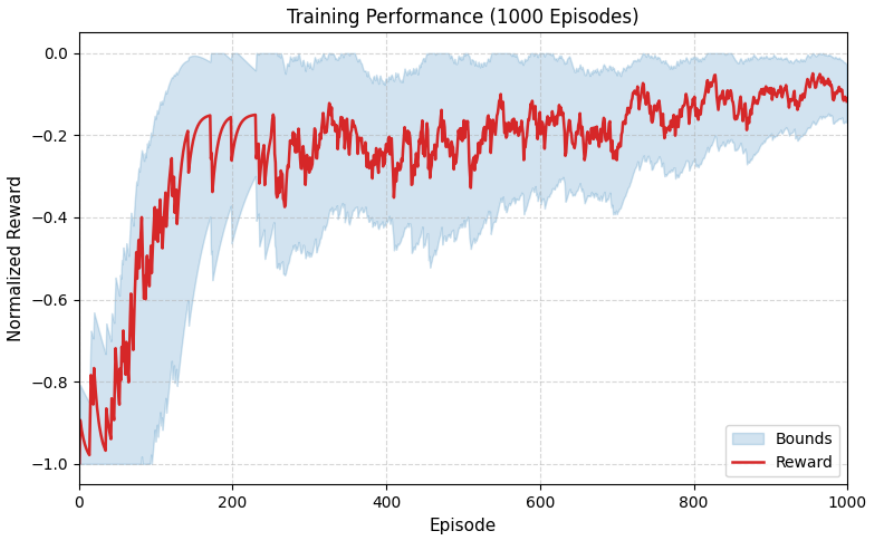

Figure 2: Average Episodic Reward During DQN Training

Figure 2 shows that the agent's average reward increases over training. This indicates that the agent is learning to adjust WIP more effectively to track the desired throughput, even when the system is uncertain and the target changes.

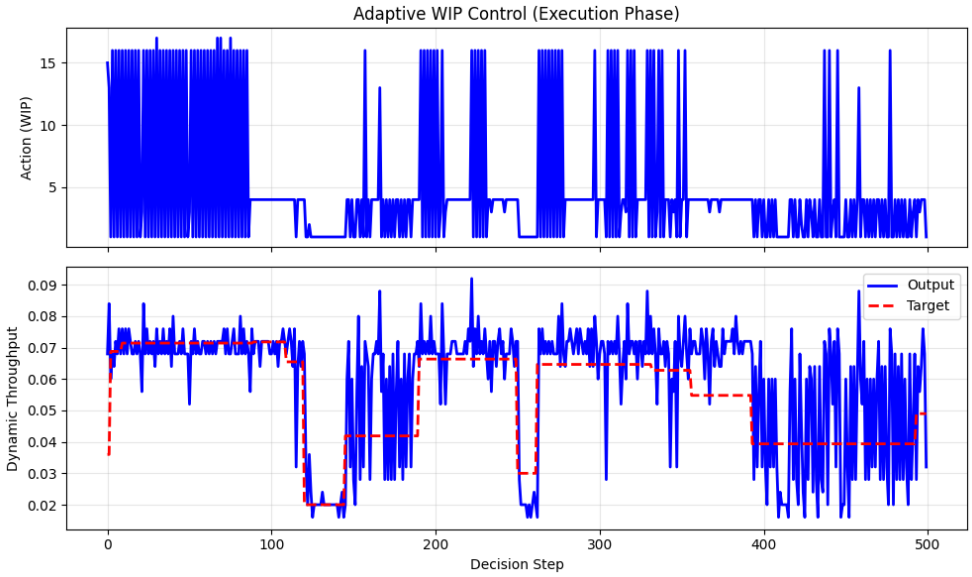

Figure 3: Learned WIP Adjustment and Throughput Tracking Under RL Control

Figure 3 shows how the agent adjusts WIP to track the desired throughput. As the target shifts, the agent modifies WIP smoothly, and the resulting throughput exhibits corresponding changes while remaining slightly below the target due to randomness in processing times and conservative WIP constraints.

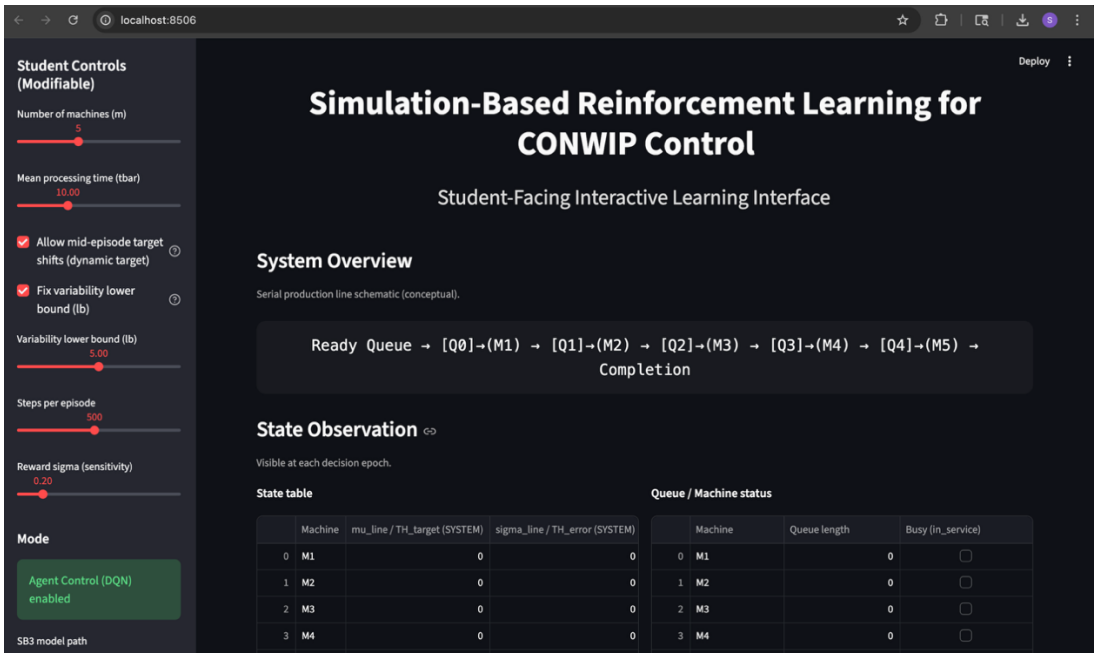

Figure 4: GUI in Proposed Environment for System Configuration

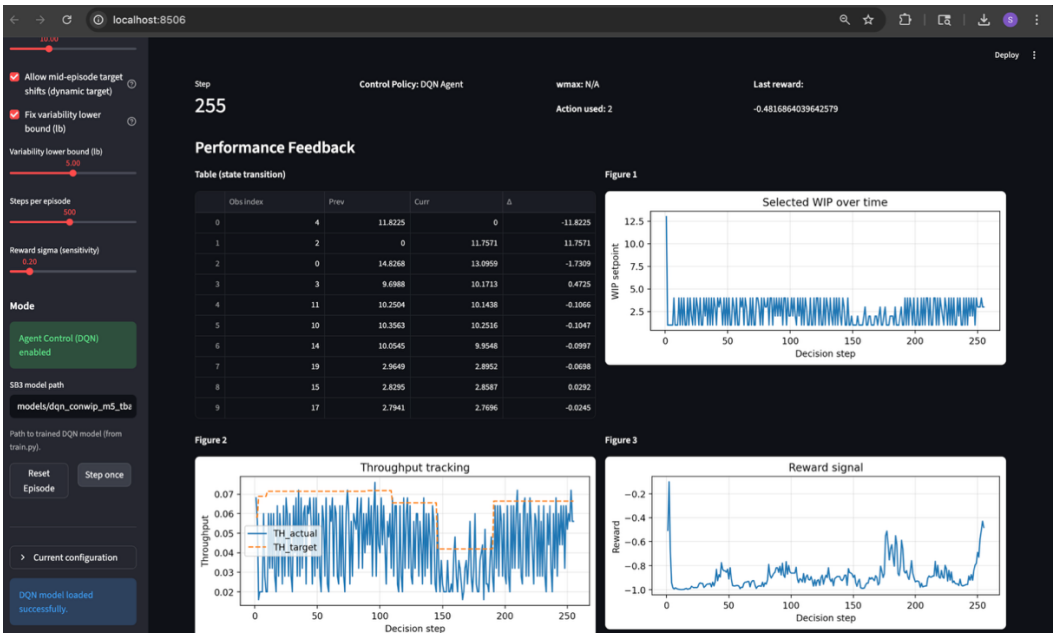

Figure 5: GUI in Proposed Environment for System Behavior Visualization

Figures 4 and 5 illustrate the GUI for system configuration and system feedback during RL control. Figure 4 presents the interface for configuring system settings, visualizing the production line structure, and monitoring key system states such as queue lengths and machine status. Figure 5 shows the evolution of system performance over time, including WIP levels, comparisons between actual and target throughput, and the resulting reward signal. Together, these interfaces support intuitive visualization of how RL decisions influence system behavior and performance within a SE context.

3.4 Comparison with Standard OpenAI Gym Benchmarks from a Systems Engineering Perspective

Standard OpenAI Gym environments such as CartPole, MountainCar, and LunarLander are effective for teaching core RL algorithms, including value updates, exploration strategies, and policy improvement. However, these environments present fully defined control problems where states, actions, and rewards are already specified. Students primarily learn how algorithms work. In contrast, the proposed manufacturing environment requires students to think about how to define system states, select meaningful control actions, and design rewards that reflect engineering objectives. Table 5 shows how it shifts learning from purely algorithm implementation toward understanding RL as a system modeling and decision-making framework within a SE context.

Table 5: Comparison of Educational Affordances

Dimension	OpenAI Gym Environment	Proposed Manufacturing Simulation Environment
Educational Emphasis	Algorithm implementation and benchmarking under predefined problem settings	Integration of algorithm implementation with system modeling and decision formulation
System Structure Visibility	Hidden dynamics with limited exposure to system structure	Explicit representation of production lines and operational flows
Problem Formulation Responsibility	State, action, and reward definitions typically predefined	Students define and modify state variables, action constraints, and reward structures
Operational Constraints	Limited real-world constraints; actions often simplified	Incorporates capacity limits, release policies, congestion effects, and control constraints
Planning vs. Learning Context	Transition models typically hidden or not emphasized	Supports discussion of known or unknown dynamics for planning and model-free methods
Engineering Context	Primarily abstract control or navigation problems	Directly grounded in manufacturing and systems engineering applications

4 Instructional Integration and Course Alignment

4.1 Alignment with RL Curriculum

The proposed manufacturing system simulation environment is designed to align with RL curriculum without requiring fundamental changes to existing course structures. Rather than

serving as a replacement for commonly used benchmark environments, the environment complements them by providing a consistent engineering system context through which core RL concepts can be illustrated and explored. This design allows instructors to incorporate the environment incrementally, using it alongside traditional examples to emphasize how RL methods apply to complex engineering systems.

Table 6 summarizes how the proposed environment can be aligned with an RL course structure. Across different topics, the same underlying simulation environment can be reused to illustrate multiple RL concepts, enabling students to revisit familiar system dynamics while focusing on different aspects of learning and decision-making. Early topics can emphasize problem formulation and interpretation of Markov decision processes, while later topics can leverage the environment to demonstrate differences among planning-based methods, model-free learning approaches, and policy-based techniques. In addition to supporting algorithm-focused topics, the environment allows students to investigate how alternative definitions of system state, control actions, and reward structures for the same physical system can lead to different learning behaviors and performance outcomes. This capability supports instructional discussion of system representation as an integral part of applying RL in engineering settings.

Table 6. Alignment of the Proposed Simulation Environment with RL Curriculum

RL Class Topics	**Instructional Use of Simulation Environment**
Introduction to Reinforcement Learning	Demonstrate sequential decision-making by controlling manufacturing operations over time and observing how short-term actions affect long-term system performance metrics.
Markov Decision Process	Discuss system states, control actions, state transitions, and the Markov property using observable manufacturing variables such as buffer levels, machine status, and processing times.
Planning by Dynamic Programming	Apply planning-based methods to simplified manufacturing configurations with known transition dynamics to illustrate value iteration and policy iteration.
Model-Free Prediction	Estimate state-value or action-value functions from simulated manufacturing trajectories under fixed control policies
Model-Free Control	Learn control policies for manufacturing systems with unknown or stochastic dynamics using trial-and-error interaction
Value Function Approximation	Apply function approximation methods to handle large state spaces arising from multiple machines, buffers, and stochastic processing times
Policy Gradient Methods	Optimize stochastic control policies for manufacturing decisions under alternative reward formulations and action constraints

Integrating Learning and Planning	Compare planning-based and learning-based approaches within the same manufacturing system under varying levels of model knowledge
Exploration and Exploitation	Examine how exploratory control actions affect manufacturing performance and learning efficiency under operational constraints
Engineering Applications	Investigate how different definitions of observable states, control actions, and reward functions for the same manufacturing system led to different learning behavior and control performance

4.2 Role of the Simulation Environment in Instruction

Within an instructional setting, the proposed simulation environment is designed to serve as a shared system context through which students can progressively engage with RL concepts [18]. Early in a course, the environment can be used for instructor-led demonstrations or guided exploration to help students interpret fundamental ideas such as sequential decision-making, system state evolution, and the Markov property in an engineering setting. At this stage, interaction with the environment emphasizes observation and reasoning about system behavior rather than algorithm implementation.

As the course advances, the same environment can support more active student engagement through experimentation and project-based exploration. Students may interact with the simulation to test different learning methods, compare planning-based and model-free approaches, and investigate how alternative modeling choices, including state representation, action constraints, and reward definitions, affect learning behavior and system performance. By revisiting a familiar system under different learning objectives, the environment enables students to connect RL algorithms to system-level outcomes and to develop intuition about how modeling and decision-making choices influence control performance in complex engineering systems.

5 Conclusion and Future Works

This paper presented an RL-enabled manufacturing system simulation environment designed to bridge RL concepts with systems engineering applications. By providing a configurable system with flexible state representations, control actions, and reward formulations, the environment supports exploration of both RL algorithms and the underlying system modeling choices that shape decision-making behavior. The educational motivation, system design, and instructional integration discussed in this paper highlight how a realistic engineering system can be used to complement standard benchmark environments and help students connect RL theory with system-level reasoning in complex, uncertain settings.

Future work will focus on deploying the proposed environment in a RL course as a project-based instructional component. The environment is being developed in collaboration with a faculty member who teaches a RL course, and planned classroom use will enable systematic evaluation of how students engage when applying RL in an engineering context. Future studies will examine instructional effectiveness through course projects, student feedback, and learning artifacts, with the goal of refining both the environment and its integration into RL education.

References

[1] M. Han, L. Yun, and L. Li, "Deep reinforcement learning-based approach for dynamic disassembly scheduling of end-of-life products with stimuli-activated self-disassembly," *J. Clean. Prod.*, vol. 423, p. 138758, Oct. 2023, doi: 10.1016/j.jclepro.2023.138758.

[2] E. Birişçi and L. Yun, "Transferable Deep Reinforcement Learning for Adaptive Control Across Varying Manufacturing System Configurations," presented at the IISE Annual Conference. Proceedings, Institute of Industrial and Systems Engineers (IISE), 2025, pp. 1–6.

[3] L. Yu *et al.*, "Deep Reinforcement Learning for Smart Home Energy Management," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 2751–2762, Apr. 2020, doi: 10.1109/JIOT.2019.2957289.

[4] A. T. D. Perera and P. Kamalaruban, "Applications of reinforcement learning in energy systems," *Renew. Sustain. Energy Rev.*, vol. 137, p. 110618, Mar. 2021, doi: 10.1016/j.rser.2020.110618.

[5] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*, vol. 1, no. 1. MIT press Cambridge, 1998.

[6] A. Zai and B. Brown, *Deep reinforcement learning in action*. Manning, 2020.

[7] G. Brockman *et al.*, "OpenAI Gym," Jun. 05, 2016, *arXiv*: arXiv:1606.01540. doi: 10.48550/arXiv.1606.01540.

[8] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–33, Feb. 2015, doi: 10.1038/nature14236.

[9] L. Yun, D. Wang, and L. Li, "Explainable multi-agent deep reinforcement learning for real-time demand response towards sustainable manufacturing," *Appl. Energy*, vol. 347, p. 121324, Oct. 2023, doi: 10.1016/j.apenergy.2023.121324.

[10] D. Reda, T. Tao, and M. van de Panne, "Learning to Locomote: Understanding How Environment Design Matters for Deep Reinforcement Learning," in *Motion, Interaction and Games*, Oct. 2020, pp. 1–10. doi: 10.1145/3424636.3426907.

[11] M. F. Rahman, A. Akundi, J. Sultana, T.-L. B. Tseng, A. Lopes, and S. Luna, "Exploring Systems Performance Using Modeling and Simulation–Project-based Study and Teaching," 2023.

[12] A. Riedmann, P. Schaper, and B. Lugrin, "Reinforcement Learning in Education: A Systematic Literature Review," *Int. J. Artif. Intell. Educ.*, vol. 35, Jul. 2025, doi: 10.1007/s40593-025-00494-6.

[13] "Introduction to Systems Engineering | Wiley," Wiley.com. Accessed: Jan. 16, 2026. [Online]. Available: https://www.wiley.com/en-us/Introduction+to+Systems+Engineering-p-9780471027669

[14] T. Lei, T. Sellers, C. Luo, Z. Bi, and G. E. Jan, "Developing Computational Intelligence Curriculum Materials to Advance Student Learning for Robot Control and Optimization," presented at the 2024 ASEE Annual Conference & Exposition, 2024.

[15] “CS 7642: Reinforcement Learning | Online Master of Science in Computer Science (OMSCS).” Accessed: Jan. 16, 2026. [Online]. Available: https://omscs.gatech.edu/cs-7642-reinforcement-learning?utm_source=chatgpt.com

[16] Z. Martin, A. Olsen, and K. Karami, “Control System Design for a Small-Scale Radio Telescope: A Senior Design Project,” presented at the 2024 ASEE Annual Conference & Exposition, 2024.

[17] S. Vespoli, G. Mattera, M. G. Marchesano, L. Nele, and G. Guizzi, “Adaptive manufacturing control with Deep Reinforcement Learning for dynamic WIP management in industry 4.0,” *Comput. Ind. Eng.*, vol. 202, p. 110966, Apr. 2025, doi: 10.1016/j.cie.2025.110966.

[18] T.-L. Tseng *et al.*, “Embedding Computer Simulation Based Classroom Activities to Enhance the Learning Experience for Manufacturing Systems,” *2020 ASEE Virtual Annu. Conf. Content Access Virtual Line*, Jun. 2020, doi: 10.18260/1-2--34519.