ZENITH STUDY HUB

Modeling and Simulation

UNIT V | ROBOTIC SYSTEMS

Manufacturing, Automation and Digital Factories

SPPU TE Robotics & Automation | 2024 Pattern | PEC321-ROA (B)

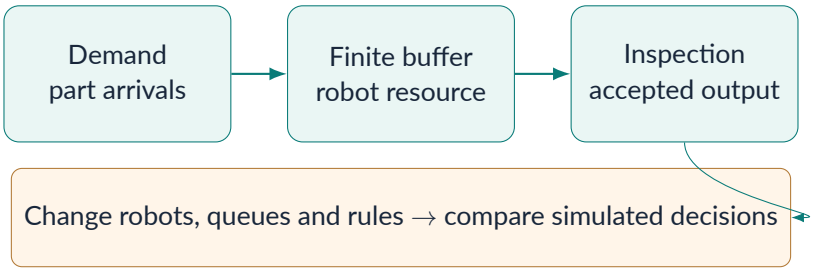

Figure 1. Original conceptual path from modeled demand to operating decisions; real-world calibration is a separate requirement.

Learning scope
This premium unit uses the official SPPU Unit V syllabus (physical PDF p. 27), original diagrams, a hand-solvable robotic-workcell example and a Python event-driven experiment with 12 reproducible replications per scenario. All times, failures and demands in the examples are hypothetical, not plant measurements.

Created: 21 November 2026 **Edition:** Learning review v1 **Audience:** TE Robotics & Automation

Unit V | Exact Syllabus Checklist

Official syllabus area	Learning focus / where to find it
Need of simulation in robotics and automation	Motivation, design decisions, safety boundaries
Product, resources, demand, control	Entities, attributes, queues, state, resource contention
Downtime, failures, random events	Failure/repair events, available time, stochastic inputs
Rework and reentrancy	Rework paths, return visits, yield definitions
Performance measures	Throughput, utilization, WIP, delay, quality, uncertainty
Material-handling systems	Transport resources, loading, finite buffers
Automated conveyors	Speed, travel time, accumulation and blocking
AGVs and mobile robots	Dispatch, route congestion, charging, deadlocks
Job shop and flexible manufacturing systems	Routing, multiple resources, setups, product families
Digital factory concepts	Physical data exchange, validation and lifecycle updating
Simulation software tools	Spreadsheet, discrete-event, robotic physics, SimPy
Case study: robotic manufacturing system	Five-job hand trace, 12-run scenarios and results

Reading convention
Every numbered figure is an original illustration. *Observed plant data* are not included. A formula that yields an ideal upper bound is never labelled as measured production. Simulation results from the accompanying script are labelled *synthetic scenarios*; the numerical comparison deliberately omits inspection/rework/failure events and treats them as documented model extensions.

1 Why Simulate Robotics and Automation?

1.1 The engineering question comes before the animation

A robotics cell is a coupled system of material arrivals, mechanical work, logic, sensors, people, constraints and uncertainty. Even when a robot's programmed cycle time is known, a production line's hourly output depends on material availability, equipment failures, synchronization and downstream capacity. Simulation allows repeatable experiments on a *defined model* before changing a real factory.

Suitable questions include: Does adding a second robotic loader reduce waiting enough to justify its cost? How many waiting slots prevent upstream blocking? Which AGV dispatch rule lowers missed material deliveries? How sensitive is throughput to repair time or part mix? These are questions about interactions, not just robot motion.

In simple language
A robot may process a part in four seconds but wait for a fixture to become free. Counting only the robot's four seconds can overstate the whole cell's production. A simulation must model the fixture if it materially changes the decision.

1.2 Simulation's limits

Simulation cannot by itself certify real collision avoidance, safety integrity, PLC function or mechanical load limits. A geometrically attractive animation can hide impossible time assumptions. Define the intended decision, process boundary, constraints, observed inputs and verification/validation tests first. Use simulation to *inform*, not replace, engineering judgment.

Five useful outputs
Good-part throughput; 95th-percentile flow time; robot utilization; queue or WIP distribution; and cost or missed-delivery rate. The correct collection depends on the decision question and safety constraints.

2 Anatomy of a Robotic Manufacturing System

2.1 Product, attributes and resources

Product entities: individually tracked workpieces, pallets, jobs or orders. Their *attributes* may include product type, priority, due time, processing route, defect state and number of processing visits. **Resources:** robot arms, grippers, operators, inspection cameras, welding fixtures, AGVs and machines, each with capacity and availability. **Demand** is the arrival process for jobs or material requests; it may be scheduled, cyclic, bursty or stochastic. **Control** is the policy that decides which task runs next, when to transport a part or which robot takes a job. An event-driven model must distinguish fixed rules from decisions being compared.

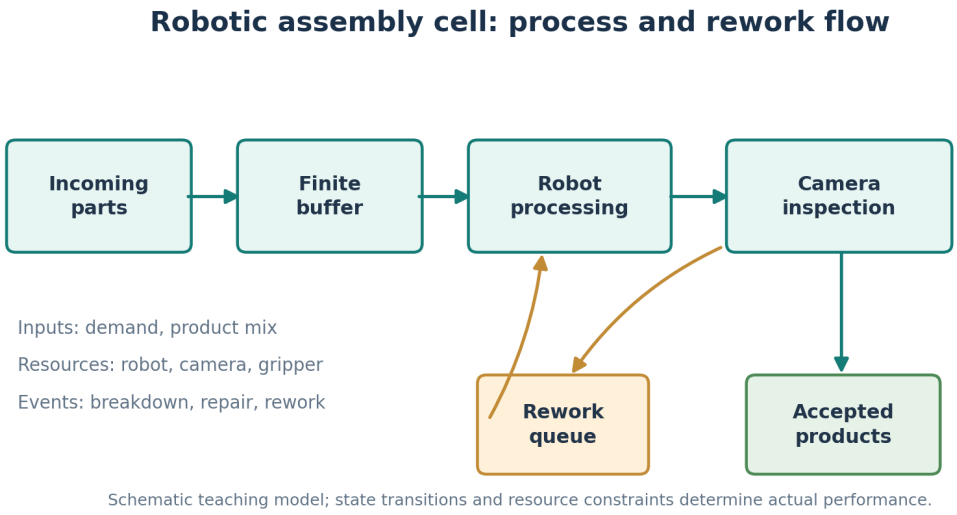

Figure 2. Original robotic workcell: parts, finite waiting buffer, robot processing, inspection and a return path for rework. The schematic is not an industrial wiring or safety drawing.

2.2 Downtime, repairs and random events

Downtime is time the required resource is unavailable. It may arise from planned maintenance or unplanned failure. Model a failure event, a repair-duration draw, the robot's availability state and the rule for work interrupted during failure. Distinguish "repair pauses work" from "failed part is scrapped";

they produce different outputs. **Random events** can include arrivals, failures, rework classification or transport delay.

Assumption trap
"95% robot uptime" is not automatically equivalent to multiplying an entire line's throughput by 0.95. Timing, buffer size, failure clustering and downstream interactions can produce a different loss. The simple availability multiplier is useful only as a stated *idealized approximation*.

2.3 Rework and reentrancy: two different ideas

Rework means a part fails an inspection or specification and requires corrective operations. A reworked part might eventually pass or be scrapped. **Reentrancy** means the job's route returns to a resource visited earlier, even when the return is part of the planned recipe (e.g., machining $\rightarrow$ inspection $\rightarrow$ same machine). A rework loop often creates reentrancy, but the terms are not interchangeable.

A **first-pass yield** denominator counts original units reaching inspection for the first time; **final yield** counts original units accepted after any allowed rework. For example, from 200 distinct original parts, if 180 pass first inspection, 8 pass after one rework and 12 are scrapped, *first-pass yield* $= 180/200 = 90\%$, *final yield* $= 188/200 = 94\%$, and *share needing rework* $= 20/200 = 10\%$. Here exactly 20 failed first inspection and 8 of those were recovered.

3 Events, States and an Executable Logic Model

3.1 What changes state, and what takes time?

An **event** is an instantaneous state change in a discrete-event model: arrival, robot-service completion, inspection classification, AGV departure or repair completion. An **activity** is an operation with duration between start and finish events. A **state variable** records the current condition, such as waiting parts $Q(t)$, busy robots $B(t)$ or robot status {idle, busy, failed}.

Events occurring at exactly the same simulated time need a deterministic tie-break policy. In the accompanying numerical model, completion events are

processed before arrivals at equal time; this ensures the newly freed robot can take an arriving part without artificially adding a wait.

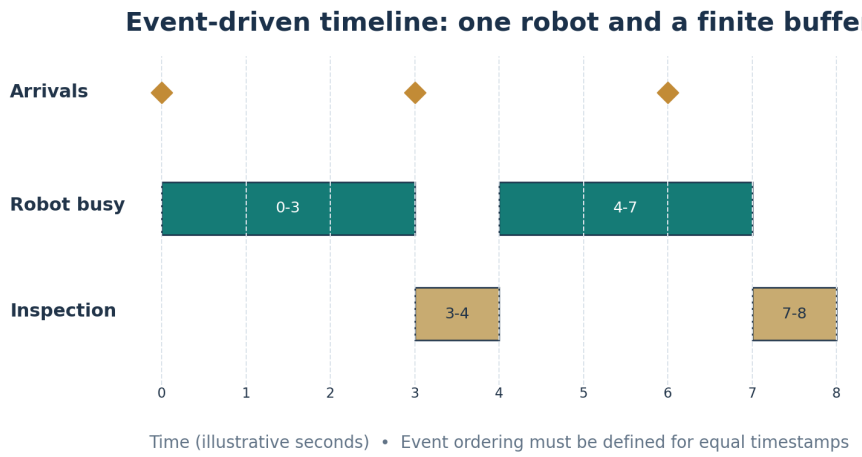

Figure 3. Original event timeline. Arrival points are instants, whereas robot service and inspection are time intervals; the difference is important for event scheduling.

3.2 State invariants (errors that software must catch)

1. $0 \leq B(t) \leq c$: no more than c robots can be busy simultaneously.
2. $Q(t) \geq 0$: a queue cannot contain a negative number of jobs.
3. Every accepted original job is either completed or still in service/waiting, when no other exit category exists.
4. Rejected/lost arrivals must be counted separately from finished-good units.
5. A robot failure must not finish a job while the machine is in the failed state unless the model explicitly defines that behavior.

Simple event loop
(1) Advance the simulation clock to the earliest event. (2) Change state according to that event. (3) Start any newly available service and schedule its future completion. (4) Update counters and the event calendar. (5) Stop at the defined horizon. Never infer durations from the number of printed animation frames.

4 Performance Measures and Their Denominators

Robotic cell performance scorecard

Throughput	Good parts / production hour	Count, time horizon
Utilization	Robot busy time / available time	Denominator policy
Mean queue delay	Average time waiting for robot	Warm-up and replications
Work in process	Parts waiting + in service	At-time or time-average
Rework rate	Units needing rework / original units	Repeat visits separate
Downtime	Unavailable resource time	Failures and planned stops

All metrics require explicit units, population, horizon and experimental conditions.

Figure 4. Original metric scorecard. It labels numerators, denominators and interpretation risks rather than showing measured industrial performance.

4.1 Throughput, utilization and waiting

Throughput: $T = N_{\text{good completed}}/\Delta t$ in good units per unit time. State whether unfinished parts at the horizon count (normally they do not). **Robot utilization:** for c identical robots each scheduled for T minutes, the fraction of *scheduled* robot capacity used is $U = \sum_r T_{r,\text{busy}}/(cT)$. If reporting utilization relative to *available* time instead, use the sum of resource-available minutes as the denominator and state that convention explicitly. **Mean queue wait:** average start time minus arrival time for a defined population, not a global time average unless derived as such. **Work in process:** count of material already admitted but not completed. A time-average WIP requires an integral over time, $\bar{L} = T^{-1}\int_0^T L(t)\,dt$. A *snapshot* of two waiting parts is not a time-average of two.

Numerical 1: robot utilization
A robot is busy for 324 minutes out of a 360-minute period in which it is available. Utilization is $324/360 = \boxed{90\%}$. If a 480-minute scheduled shift includes 120 minutes of unavailable time, a separate ratio using the whole shift is $324/480 = \boxed{67.5\%}$. Report which denominator the question requests.

4.2 Lead time and quality measures

Lead/flow time is departure minus arrival for each specified job; a 95th percentile is a tail measure, not five percent of the mean. **Rework rate** should be based on *unique original units requiring rework*, unless a separate "number of rework visits per unit" statistic is intended. A repeatedly reworked part must not silently increase the original-unit denominator.

5 Material Handling and Finite Buffers

Material handling couples workstations through forklifts, conveyors, robot transfers, pallets, operators or AGVs. A resource may have capacity one or more; loading/unloading and transit times may compete with other tasks. The model must represent **starvation** (downstream machine has no input) and **blocking** (upstream machine cannot release a completed part because the downstream location is full).

5.1 Finite buffer example

Suppose a buffer can hold two *waiting* parts and the robot is busy. A third arrival cannot join that queue. If the policy is "reject", it increments a lost-arrival counter; if the policy is "block upstream", the part remains outside the cell and an upstream resource may stop. These different physical meanings must never share a label.

Numerical 2: conservation
Start with $Q = 2$ waiting, the robot busy and a two-slot capacity. One new part arrives: with a rejection policy, Q stays 2 and lost arrivals increase by 1. At the next completion event, the robot immediately starts the oldest waiting job, so Q falls to 1. A source that reports $Q = 3$ has violated its stated capacity.

6 Automated Conveyor Systems

6.1 Travel, accumulation, blocking and spacing

A belt's speed and path length determine an *ideal transit time*. Real delivery can take longer because of merge zones, sensors, transfer handoffs and accumulation. Space between products also limits the ideal flow rate. If product pitch is p meters and belt speed v meters/second, ideal passage frequency is v/p products per second when motion and spacing remain constant.

Numerical 3: conveyor motion
With path length $L = 12$ m and speed $v = 0.4$ m/s, ideal belt travel time is

$$t = L/v = 12/0.4 = \boxed{30 \text{ s}}.$$

If pitch is 0.8 m, the ideal flow bound is $0.4/0.8 = 0.5$ parts/s, or $\boxed{1800}$ parts/hour, assuming uninterrupted spaced flow. This does not imply a robot on the line can process 1800 parts/hour.

6.2 Where a conveyor model can be wrong

Treating a moving belt as a zero-time connection can misstate WIP and response time. Modeling every bearing in detail is unnecessary for a coarse throughput decision. Use a higher-fidelity physical model when acceleration, collision or line tracking materially changes the question.

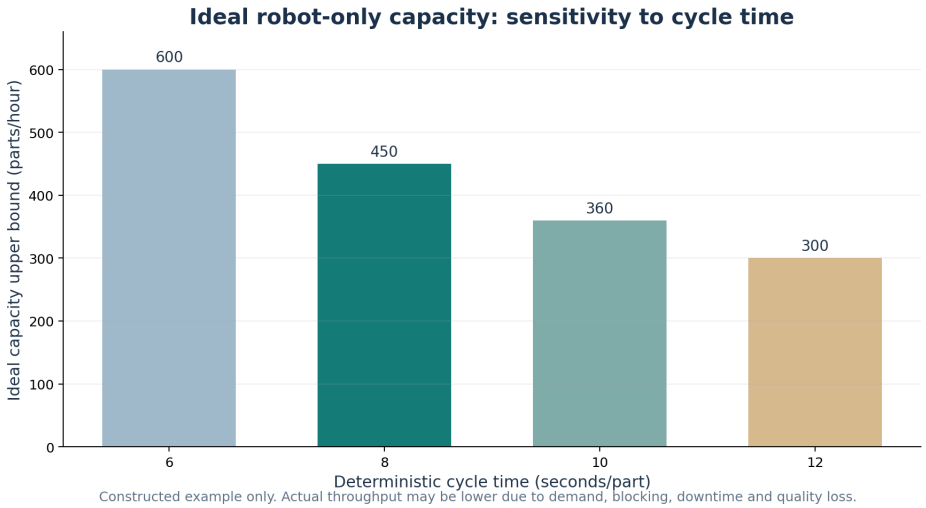

Figure 5. Original theoretical single-robot capacity versus fixed cycle time. It is an ideal upper bound for one continuously supplied, loss-free workstation, not factory throughput data.

Numerical 4: bottleneck sensitivity
In a loss-free single-robot cell, $R_{\max} = 3600/c$ parts/h for cycle time c in seconds. For $c = 6, 8, 10, 12$ s, the respective bounds are $\boxed{600, 450, 360, 300}$ parts/hour. Reducing conveyor transit time cannot raise overall output beyond the robot's cycle-time bound if the robot remains the limiting step.

7 AGVs and Autonomous Mobile Robots

7.1 Why logistics deserves its own model

An automated guided vehicle (AGV) or autonomous mobile robot (AMR) is a shared transport resource. Its travel time, payload, charging time, priority rules and conflicts with other vehicles can become the bottleneck even when the robot arm is fast. An AGV may follow a fixed guide path; an AMR often provides more flexible localization and path selection, but those product labels alone do not prove any safety or performance property.

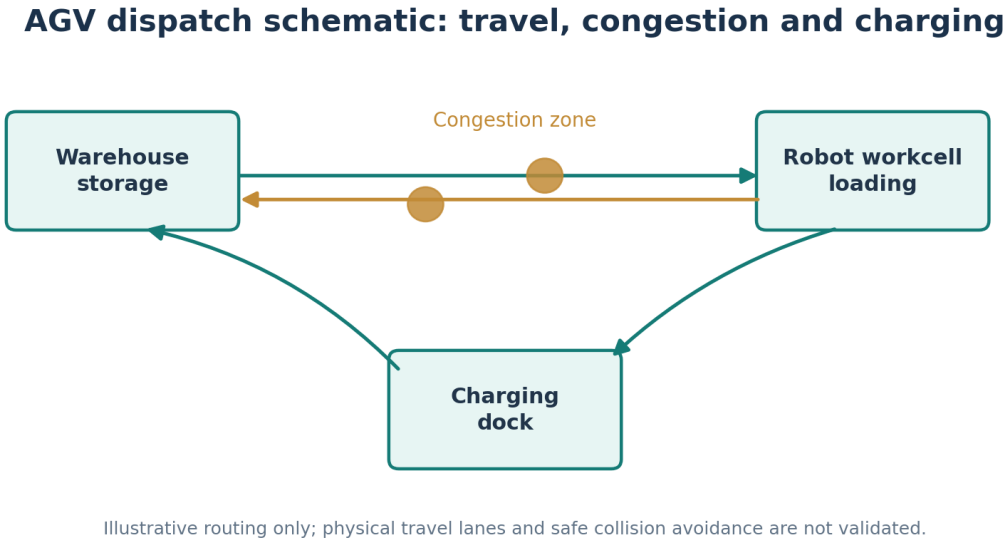

Figure 6. Original layout schematic showing transport demand, dispatch, stations, charging and possible corridor contention. Routes are illustrative, not mapped or safety-validated paths.

7.2 Route and dispatch logic

A useful transport model records request generation time, pickup and delivery station, empty-travel time, loaded-travel time, loading/unloading, waiting for a destination and charging. Compare dispatch strategies: first-request-first-served, nearest eligible vehicle, earliest due job or workload balancing. A "nearest" rule may be inferior if a nearer vehicle has too little charge or its route is congested.

Numerical 5: ideal AGV travel
A 60-meter route traversed at a constant 1 m/s takes $60/1 = \boxed{60}$ seconds *of driving*. If loading takes 12 seconds and unloading 8 seconds, the minimum handling-inclusive time is $\boxed{80}$ seconds, before turns, stops, queues, congestion and charging. State the assumptions if using this as a transport service time.

7.3 Congestion, deadlock and charging

When multiple vehicles compete for a one-way aisle, a shared segment can be modeled as a capacity-one resource. Without an explicit right-of-way or reservation rule, a circular waiting condition can create deadlock. Charging consumes time and a physical charger resource; omitting it can overstate sus-

tained fleet utilization. Path feasibility must still be tested with the actual vehicle, safety scanners and site layout.

Safety boundary
A diagram route or simulated traffic policy is not an approved mobile-robot safety system. Human-robot separation, stopping distance and applicable standards require a distinct physical risk assessment and testing.

8 Job Shops and Flexible Manufacturing Systems

8.1 Fixed line versus job shop

In a fixed production line, most jobs follow the same ordered route, for example cutting → drilling → inspection. A **job shop** supports different job-specific sequences, such as Product A: M1 → M2 → Robot, and Product B: M2 → Robot → M1. Resource queues at M1 and M2 are shared, so choosing dispatch policies changes completion and due-date performance.

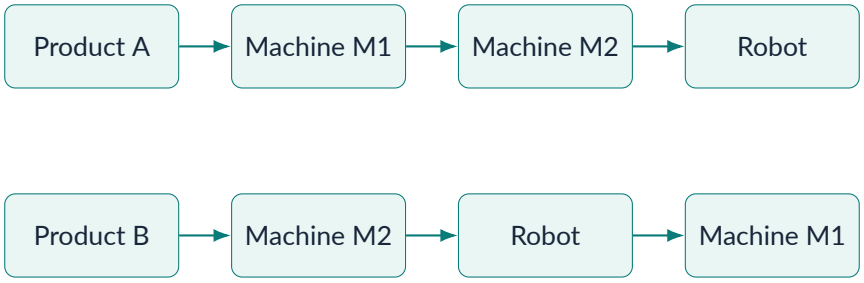

Figure 7. Original job-shop routing example: different products visit common resources in different orders. Each processing stage creates possible contention.

8.2 Dispatching rules

FIFO picks the oldest waiting job. **Shortest processing time** picks the shortest available job; it may reduce mean flow time but delay long jobs. **Earliest due date** ranks by deadline and can help certain lateness objectives, without guaranteeing a global optimum. Comparisons must use the same arrivals, routes, resource capacities and setup-time assumptions.

Numerical 6: one resource, two scheduling choices
Suppose two jobs are ready at time zero: Job A requires 9 minutes and B requires 3 minutes. Under FIFO with A first, completion times are 9 and 12 minutes, mean $\boxed{10.5}$ minutes. Under shortest-processing-time first, B finishes at 3 and A at 12 minutes, mean $\boxed{7.5}$ minutes. Neither rule is automatically preferable for due dates or fairness; the chosen objective matters.

8.3 Flexible manufacturing systems (FMS)

An FMS integrates configurable machines, tools, robots, routing and computer control to process different product families. Simulation can ask which fixture, robot, tool magazine or AGV constrains a proposed mix. **Setup times** between product families and breakdowns often matter as much as nominal processing capacity. **Re-entrant visits** to a shared machine are especially important because one job can cause multiple queueing encounters.

9 Digital Factory Concepts and the Digital Twin

9.1 What a digital twin adds

An offline simulation model can answer a hypothetical design question without any live connection. A **digital factory** combines appropriate digital descriptions of production assets, processes, layouts and operations. For a *digital twin* of a working robot cell, a maintained connection to relevant physical-system data and an explicitly validated representation are important. NIST research uses manufacturing digital-twin frameworks and standards including ISO 23247; the exact architecture depends on the purpose and information flows.

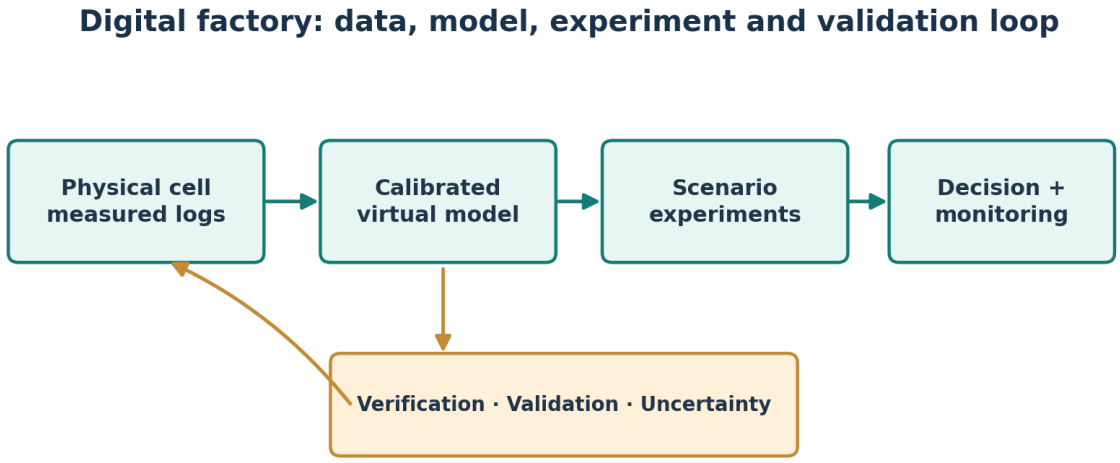

Figure 8. Original verification/validation feedback schematic for a data-informed robotic work-cell model; calibration and uncertainty checks connect modeling with the actual process.

9.2 Typical information flows

Sensor and controller data may provide robot status, cycle timestamps, job IDs, rejection events, fault codes and tool-change history. The model maps these measurements to recognized states; calibration and validation check that the modeled behavior is sufficiently close for the intended task. A digital twin used for scheduling may not require the same geometric precision as one used for joint-level collision or wear studies.

Avoid a misleading claim
A 3D CAD animation, a historical KPI dashboard and a standalone discrete-event model are useful, but none automatically becomes a high-fidelity digital twin simply because its visual appearance resembles the factory.

10 Choosing Simulation Software Tools

Tool class	Typical engineering question	Strength / caution
Spreadsheets	Simple capacity or small queue trace	Fast and auditable; awkward for many events
Discrete-event engines	Resources, queues, jobs, buffers	Time/event logic; limited joint physics
SimPy (Python)	Custom process-based DES experiments	Readable Python; modeler's logic must be verified
Commercial DES tools	Large manufacturing logistics studies	Integrated visualization; license/validation still matter
Robotics or physics simulators	Robot trajectories and geometric interaction	Needs sound geometry, dynamics and controller assumptions
Hybrid co-simulation	Discrete flow coupled to continuous robot dynamics	Interfaces and time synchronization require special care

10.1 SimPy as an example of an event-oriented library

SimPy provides `Environment`, processes, timeout events and shared `Resource` objects. A Python generator can request a resource, wait for its availability, simulate service using an environment timeout and release the resource. This is suitable for a queueing cell, but does not automatically perform physical collision checking. Use the official SimPy documentation for the exact installed API version.

Listing 1: Original illustrative SimPy process sketch; requires SimPy installed

```
def process_part(env, robot, service_time):
    arrival_time = env.now
    with robot.request() as request:
        yield request
        waiting_time = env.now - arrival_time
        yield env.timeout(service_time)
    return waiting_time
```

Runnable accompanying verification
The deliverable also includes `workcell_verification.py`, a self-contained Python event-scheduling experiment (using Python's standard event-queue logic and SciPy only for confidence intervals). The numerical case-study tables below were generated from this script, not guessed. The SimPy sketch above is explanatory and was not represented as a run of the provided script.

11 Worked Case Study: One Robot or Two?

11.1 Study question, scope and inputs

Question: How would an additional robot or a short waiting buffer affect an artificial assembly-handling cell during one 480-minute shift? We study arrival times, robot services, queue delays and rejected arrivals. *This numerical experiment does not include inspection durations, rework passes or failures*; those event types are explained in the earlier workcell chapters and must be added to predict a real plant.

Input	Simulation convention
Arrival process	Exponential interarrival times, mean 4.0 minutes
Robot service	Triangular service-time distribution: min 3.5, mode 4.5, max 6.5 min
Time horizon	480 minutes (one terminating synthetic shift)
Starting condition	Empty queue; robots idle; no warm-up deletion
Robots	One or two identical parallel servers
Queue discipline	FIFO; completion events precede arrivals if times tie
Waiting capacity	Unlimited or three waiting slots (not counting in-service parts)
Rejection convention	If full, arriving original job is rejected and recorded
Independent runs	12 replications, one seed per replication (shown in source)
Scenarios	Identical sampled arrivals and service requirements within each replication

11.2 Hand-solvable five-job baseline

Before using stochastic draws, verify the simplest possible deterministic event trace. Let five jobs arrive at $t = (0, 2, 6, 8, 13)$ minutes and require service times

$(4, 3, 5, 2, 4)$ minutes. One robot, FIFO and no failures yield the table below.

Job	Arrive	Service	Start	Finish	Wait (min)
1	0	4	0	4	0
2	2	3	4	7	2
3	6	5	7	12	1
4	8	2	12	14	4
5	13	4	14	18	1

Numerical 7: deterministic queue test
Total waiting $= 0 + 2 + 1 + 4 + 1 = 8$ minutes, average $= 8/5 = \boxed{1.6}$ min. Robot is busy for $4{+}3{+}5{+}2{+}4 = \boxed{18}$ min from $t = 0$ to 18, so this *particular* observation window has utilization 100%. All five jobs complete. This is a hand-verifiable software oracle, not a long-run performance prediction.

11.3 Why adding a robot changes the timeline

With two otherwise identical idle robots, all five jobs can start immediately at arrival. Their completion times are 4, 5, 11, 10 and 17 minutes by job ID, and their waits are all zero. If measured from 0 to 18 minutes, the two robots together are busy for 18 robot-minutes out of $2 \times 18 = 36$ potential robot-minutes, so combined utilization is 50%. Two robots reduce this example's waiting but *increase idle capacity*; economic justification is a separate question.

Correct accounting identities
At an observation horizon with no rework or outside transfer, *accepted originals* $=$ *completed originals* $+$ *unfinished WIP*. Separately, *arrivals* $=$ *accepted* $+$ *rejected*. The verification script asserts both identities for every event trace.

11.4 A stochastic shift experiment

The published `workcell_verification.py` generates exponential arrivals and triangular services and compares three designs. Each replication uses a distinct fixed seed, and the *same input realization is applied across the three designs* within each replication (common random numbers). For each design, the printed 95% confidence interval estimates the mean shift completion count from 12 replication-level outputs; it does not bound all individual shifts or model uncertainty.

Metric	A: 1 robot, unlimited	B: 2 robots, unlimited	C: 1 robot, 3 slots
Mean completed / shift	96.25	115.25	90.75
95% CI for mean completions	(94.26, 98.24)	(109.64, 120.86)	(88.08, 93.42)
Mean rejected originals	0	0	23.50
Mean busy fraction	0.9698	0.5807	0.9147
Mean delay of started jobs (min)	41.54	1.30	7.72

All entries are *recomputed synthetic outputs* from 12 replications of the model defined above. "Mean busy fraction" uses the total scheduled robot capacity $480 \times c$ robot-minutes as denominator, and not a hardware availability-adjusted denominator. The queue-delay metric averages jobs that *started service* by the observation horizon, not all arriving jobs; the finite-buffer scenario rejects some arrivals rather than waiting for them.

A fairness warning for finite buffers
Scenario C has a shorter mean admitted-job wait than A partly because it *rejects* many arrivals. A recommendation based only on waiting time would therefore be misleading. Compare the lost-arrival rate and successful output alongside delay and robot utilization.

11.5 Interpreting the scenarios

Scenario A is heavily loaded and accumulates queue delay. Scenario B provides extra capacity, improving mean completed throughput and lowering waiting in these synthetic tests, at the cost of additional resources. Scenario C caps waiting but rejects incoming jobs; the lower mean queue delay must not be interpreted as a universally better production policy. A purchase decision would need measured process inputs, equipment cost, safety constraints, maintenance requirements and validated uncertainty estimates.

It is incorrect to subtract the individual two-sample confidence interval endpoints and report that as a confidence interval for the improvement. Because the study applies common random numbers *within each replication*, it analyzes the paired completion difference (in parts) for B minus A:

$$D = (23, 16, 35, 20, 27, 14, 7, 26, 3, 25, 10, 22).$$

The mean difference is $\bar{D} = \boxed{19.00}$ parts/shift, with sample standard deviation $s_D = 9.2736$. A 95% Student-t interval over 12 independent *pairs* is

$$19.00 \pm t_{0.975,11}\frac{9.2736}{\sqrt{12}} = \boxed{(13.11, 24.89) \text{ parts/shift}}.$$

This interval measures Monte Carlo uncertainty *conditional on the synthetic input model*; it does not quantify physical plant model error. Unit IV supplies the general paired-difference method.

12 Extending the Cell: Inspection, Rework and Failures

12.1 Inspection and repeated visits

For each completion, schedule inspection. If the job fails, update the original part's status and re-enter the robot queue using a stated rule. Count robot processing *visits* separately from *original parts*. If a job returns twice, counting it as three new original arrivals would inflate demand and distort yield. The stochastic table above intentionally omits these return visits; a production simulation must add them if important.

Numerical 8: first-pass versus final yield
For 200 original parts, 180 pass first inspection, 20 enter one rework, 8 of the reworked pass and 12 are scrapped. First-pass yield $180/200 = \boxed{90\%}$. Final yield $(180 + 8)/200 = \boxed{94\%}$. The rework-required rate is $20/200 = \boxed{10\%}$. Robot service visits total 220 if every original part gets one initial robot visit and all 20 return exactly once; this is *not* 220 distinct original products.

12.2 Failure-event rules

If the robot fails during a service, specify whether work resumes with remaining time, restarts from the beginning or is scrapped. Model a repair crew as another potentially shared resource. Report available time separately from scheduled time. For instance, 24 minutes unavailable out of a 480-minute shift implies 95% *time availability*; it does not automatically imply 95% of ideal throughput because event timing, queues and unfinished jobs matter.

12.3 Checking a richer model

Test with zero failures, zero inspection time and zero rework first: the detailed model should reduce to the simpler verified baseline under these limits. Then enable one feature at a time, add extreme cases and compare against actual logs. This is stronger than assuming that a visually more complex digital twin must be more accurate.

13 How to Verify and Validate a Manufacturing Simulation

1. **Verify against a hand trace:** compare events, queues and completions for five deterministic jobs.
2. **Test invariants:** queue length never negative; no busy robots beyond capacity; all admitted jobs accounted for.
3. **Test sensitivity:** zero downtime, zero demand, two robots, tiny buffer and very high arrival rate.

4. **Record random streams:** seed, pseudorandom generator, model version, scenario inputs and unit conversions.
5. **Validate with real shifts:** compare modeled arrivals, processing durations, downtime, rejects and selected outputs with measurements not used solely for calibration.
6. **Quantify uncertainty:** report independent replications and a confidence interval or suitable time-series method; assess input-model sensitivity separately.
7. **Review safety externally:** do not infer safe physical operation from correct discrete-event results or a 3D animation.

What this unit proves and what it does not
It proves the documented synthetic arithmetic and implementation invariants agree with reproducible script results. It does not prove that the model predicts a particular manufacturer, that its input distributions match an actual workcell, or that robot and AGV safety requirements are met.

14 Exam Revision: Questions and Answer Cues

1. **Why simulate robotics?** To compare coupled resources, uncertainty and control alternatives before physical changes.
2. **Define product and resource.** Product is an individual job/part with attributes; resource is constrained handling/processing capacity.
3. **Distinguish demand and control.** Demand generates work; control decides how work is scheduled or routed.
4. **What is an event?** Instantaneous state change at a simulated time point.
5. **Name key manufacturing events.** Arrival, service start/completion, inspection, failure/repair, transport and rework.
6. **Rework versus reentrancy?** Corrective processing versus a route returning to an earlier resource.

7. **What is blocking?** Upstream completion cannot hand off because downstream space is occupied.
8. **What is starvation?** A resource waits because there is no material to process.
9. **Give a finite-buffer invariant.** Queue occupancy does not exceed its declared capacity.
10. **State throughput formula.** Completed good original units divided by the stated measurement horizon.
11. **State utilization denominator.** Busy time divided by explicitly defined available or scheduled resource time.
12. **Explain work-in-process.** Accepted but unfinished jobs; specify snapshot or time-average.
13. **Conveyor numerical:** 12 m at 0.4 m/s $\rightarrow$ 30 s ideal transit.
14. **AGV numerical:** 60 m at 1 m/s $\rightarrow$ 60 s ideal driving.
15. **How is FMS different from a fixed line?** Multiple products, routing options and reconfigurable resources.
16. **Compare FIFO and SPT.** FIFO serves earliest arrival; SPT selects shortest task, favoring certain mean-delay objectives.
17. **Why model charging?** It removes mobile resources from transport duty and can form a shared bottleneck.
18. **What is digital twin validation?** Checking data-informed representation against physical evidence for a stated decision use.
19. **What does SimPy offer?** Python discrete-event processes, resources, events and timeouts.
20. **Why do multiple replications?** To quantify random output variability, rather than trust one shift simulation.
21. **Explain why a capped queue can look misleadingly fast.** It may reject jobs rather than serve them.
22. **What is the limitation of the Unit V numerical study?** No calibrated real plant data and no inspection/rework/failure events in the numerical comparison.

Sources, Original Figures and Verification Boundary

Official syllabus: SPPU, Third Year Robotics and Automation, 2024 Pattern, PEC321-ROA (B), Unit V, physical PDF page 27. The exact list of topics and course objectives was directly checked.

Available course materials: The linked *ENDSEM NOV 2026* Drive collection contains the M&S Units I and II lecture PDFs, both question banks and the SPPU syllabus. These files were opened and their contents read. No Unit V-specific college lecture PDF was found in the inspected folders; Units I–II materials are not misrepresented as Unit V evidence.

Textbook reference: Manuel D. Rossetti, *Simulation Modeling and Arena*, open third edition, available locally; used for modeling vocabulary and the general simulation workflow. The official syllabus also lists Averill Law and Banks et al. as reading recommendations, but their images or text are not reproduced.

Verified public research: NIST: Towards a Digital Twin of a Robot Workcell; NIST: Digital Twins for Advanced Manufacturing; SimPy documentation; Winter Simulation Conference: verification/validation review.

Visuals: Six original labelled PNG diagrams prepared for this unit, plus an original TikZ job-shop routing diagram and cover illustration. Editable source graphics are included in the accompanying bundle; no third-party photo or textbook image is copied.

Quality boundary: This study guide was checked by compilation, arithmetic scripts, event-model invariants and full PDF page rendering. It remains a learning-review edition; independent academic review, industrial-data calibration, physical robot safety validation and live website publication are separate release gates.

UNIT V | P0 syllabus-gap corrections

Supplement to the existing reviewed edition. All original teaching pages remain unchanged. The material below closes only the specifically rejected syllabus/question-bank gaps.

1. Departmental Q4: MTBF, MTTR and downtime models

MTBF (mean time between failures) is the average operating interval between failure occurrences for a repairable system under a specified observation and reporting convention. MTTR (mean time to repair) is the average duration required to restore service after failure; document whether diagnostics, waiting for spares and testing are included. Both are averages, not fixed repair schedules.

For a simple alternating up/down repairable resource with independent identically distributed up and repair durations and finite means, intrinsic steady-state availability is A=MTBF/(MTBF+MTTR). Example: MTBF=120 operating hours and MTTR=6 hours gives A=120/(120+6)=0.95238, or 95.24%. This does not imply line throughput is exactly 95.24% of its ideal maximum: buffers, correlated failures and scheduling can change the effect.

Example distributions: an exponential time-to-failure model has constant hazard rate λ=1/MTBF, appropriate only if memorylessness is defensible. A Weibull lifetime model can represent increasing or decreasing hazard with age; repair durations may be lognormal, gamma or empirical when positive and skewed. Use maintenance logs to fit and validate rather than automatically assigning an exponential law. In a discrete-event simulation, schedule failure, pause or scrap interrupted work according to an explicit rule, draw repair time, restore the resource and track uptime, interruptions, WIP and good-part throughput.

2. Departmental Q9: Accumulating versus non-accumulating conveyors

Aspect	Accumulating conveyor	Non-accumulating conveyor
Stop behavior	Can hold or queue products on the conveyor while upstream motion may continue depending on zones and controls	Line/belt commonly stops or must prevent loading when downstream flow is interrupted
Product motion	Products may wait locally; controlled low-/zero-pressure accumulation is possible	Items tend to advance with conveyor motion or stop as a coupled flow
Key simulation state	Per-zone occupancy, capacities, release logic and product spacing	Shared drive state, travel time, entry/exit and downstream blocking
Common use	Buffer between workstations with variable cycle times	Simple transfer when synchronous flow is acceptable

Illustrative bottleneck: if downstream inspection stops for 20 seconds, a two-zone accumulating conveyor may hold two incoming parts and postpone upstream blocking until the zones fill. A non-accumulating conveyor without spare buffering must stop/restrict feed sooner. Do not claim accumulation eliminates blocking: finite capacity eventually fills. Conveyor length/speed give ideal travel time L/v, but actual delays depend on zone control, sensor positions, merging, and blocking.

Question-bank alignment: M&S Question Bank.pdf, Unit V, Q4 and Q9. Values are explicitly illustrative; not collected from a plant. The original Unit V 12-replication comparison is retained unaltered and is not claimed reproduced by these corrections.