Received: 13 Jan. 2026; Revised: 19 Feb. 2026; Accepted: 25 June 2026
https://doi.org/10.22306/atec.v12i2.318

Digital value chain with cognitive technologies: a conceptual framework and small-scale prototype

Maksym Karakai
Technical University of Košice, Faculty of Manufacturing Technologies with a seat in Prešov, Department of Manufacturing Management, Bayerova 1, 080 01 Prešov, Slovak Republic, EU, maksym.karakai@tuke.sk
Daniel Bobko
Technical University of Košice, Faculty of Manufacturing Technologies with a seat in Prešov, Department of Manufacturing Management, Bayerova 1, 080 01 Prešov, Slovak Republic, EU, daniel.bobko@tuke.sk
Reinhard Bernsteiner
MCI Management Center Innsbruck | The Entrepreneurial School, Department Management, Communication & IT, Universitaetsstrasse 15, 6020 Innsbruck, Austria, EU, reinhard.bernsteiner@mci.edu
Lucia Knapčíková
Technical University of Košice, Faculty of Manufacturing Technologies with a seat in Prešov, Department of Manufacturing Management, Bayerova 1, 080 01 Prešov, Slovak Republic, EU, lucia.knapcikova@tuke.sk (corresponding author)

Keywords: digital value chain, MQTT, OPC UA, cognitive manufacturing, discrete event simulation.
Abstract: This paper proposes a conceptual framework linking digital value chain integration with cognitive technologies and demonstrates it through a simulation-based micro-prototype. The framework combines discrete-event simulation, event-driven communication, and an external decision component to realize a minimal perception–decision–action loop. A discrete-event model was implemented in Tecnomatix Plant Simulation and connected to external components via MQTT publish/subscribe messaging. During runtime, the simulation emits process events (e.g., station start/finish) that can be observed through standard MQTT tooling. External commands are delivered through the broker to control execution (START/STOP) and to actuate a selected model parameter (processing-time updates), while a simple constraint mechanism provides alert signaling for out-of-range inputs. To support reproducibility and offline analysis, a lightweight Python MQTT bridge captures message streams and exports structured telemetry logs (JSONL/CSV). The prototype illustrates how cognitive logic can be decoupled from the simulation environment while remaining dynamically integrated through events and commands. Although the demonstrator is intentionally minimal and not intended for industrial optimization, it provides an experimentally grounded example of modular integration principles applicable to scalable and interoperable digital value chain architectures. The contribution of this work lies in demonstrating a minimal and reproducible event-driven integration pattern rather than an industrial-scale digital twin or optimization solution.

1 Introduction

Digital transformation has become a key driver of change in manufacturing systems, driven by increasing process complexity, shorter product life cycles, and growing demands for flexibility and resilience. Within this context, the concept of the digital value chain expands traditional views of value creation by emphasizing continuous digital integration across planning, execution, monitoring, and improvement. The effective use of digital data and models across these stages is widely considered a prerequisite for intelligent and adaptive manufacturing environments, yet practical adoption often remains fragmented across heterogeneous tools and interfaces, which limits end-to-end responsiveness and decision support [1].

In recent years, digital twins and simulation-based approaches have been widely discussed and adopted as core elements of digital manufacturing strategies. Discrete-event simulation, in particular, provides a controllable environment for analyzing production system behavior under varying conditions and for evaluating decisions during system design and operation. However, in many practical applications, simulation models are still applied as isolated tools—supporting descriptive or predictive analysis without seamless integration of adaptive decision mechanisms and runtime feedback loops. Consequently, their potential to contribute to autonomous or semi-autonomous behavior across the digital value chain is often underutilized [2].

At the same time, cognitive technologies and artificial intelligence are increasingly presented as enablers of intelligent manufacturing systems. These approaches range from transparent rule-based logic and agent-based decision support to data-driven learning methods. Despite their promise, many integration efforts either focus on narrowly scoped optimization tasks or rely on complex, large-scale architecture, which can reduce transparency and complicate experimental validation. This creates a clear need for simple, modular, and experimentally grounded demonstrations that show—step by step—how external decision logic can interact with digital representations of production processes in a way that remains interpretable and reproducible [3].

Event-driven architecture provides a suitable integration mechanism in this regard, enabling loosely coupled communication based on the exchange of events. By decoupling simulation from external logic, event-based integration supports modularity and scalability, and it enables perception–decision–action loops without embedding complex cognitive logic directly into the simulation environment. Lightweight publish/subscribe messaging (e.g., MQTT) further supports rapid prototyping and interoperability across distributed components, making it a practical choice for small experimental setups and for future expansion toward richer digital value chain scenarios [4].

The aim of this paper is to propose a conceptual framework that links digital value chain concepts with cognitive technologies through event-driven integration and to demonstrate its feasibility using a simulation-based micro-prototype. The prototype integrates a discrete-event simulation model in Tecnomatix Plant Simulation with external control and evidence collection via MQTT-based messaging. It demonstrates broker-mediated observability (process events), command-driven actuation (execution control and parameter updates), and structured logging through a lightweight Python bridge. By focusing on a minimal yet complete interaction loop, the paper provides a clear foundation for further research on adaptive and intelligent digital value chain systems and motivates the systematic literature positioning presented in the following section.

While digital twins, event-driven architectures, and cognitive technologies are widely discussed in the context of smart manufacturing, many existing studies either focus on large-scale and complex reference architectures or address narrowly defined optimization tasks. This often limits transparency, experimental accessibility, and reproducibility, particularly in early-stage research and methodological exploration.

The contribution of this paper lies in proposing and experimentally demonstrating a minimal yet complete integration framework that links discrete-event simulation with external cognitive logic through event-driven communication. Rather than aiming at industrial-scale optimization or a fully-fledged digital twin, this study introduces a simulation-based micro-prototype that explicitly realizes a perception–decision–action loop using lightweight and widely available technologies.

By emphasizing modularity, decoupling, and reproducibility, the presented framework provides an experimentally grounded building block for future research on cognitive and adaptive digital value chain architectures.

2 Literature overview

2.1 Digital value chain and smart factory integration

The digital value chain can be interpreted as an integrated flow of information and decisions across the lifecycle and operational chain—from planning to execution and continuous improvement—where value is created by reducing uncertainty, improving coordination, and accelerating response. Industry 4.0 literature frames smart factories as networks of connected resources and information systems; however, implementation barriers often stem from interoperability issues, uneven digital maturity, and insufficient integration between analytics and operational systems [1].

From a practical perspective, simulation and digital tools are increasingly used to analyze and improve value chain performance in Industry 4.0 settings, including supply chain and production system coordination. This supports a view where value chain "digitalization" is not a single technology but an ecosystem of methods (DES, digital twins, IIoT connectivity, analytics) that must be orchestrated [3].

2.2 Digital twins and "micro" digital twins as building block

Digital twin research ranges from conceptual definitions to detailed reference architecture. A recurring message is that effective digital twins require reliable data acquisition and connectivity, a coherent data model and persistence, and analytics/simulation components capable of generating insight and recommending actions. In manufacturing case studies, layered architectures enable separation of concerns (local data handling, gateways/connectivity, cloud databases, simulation/emulation, and application logic), which is especially relevant when integrating legacy systems and new analytics modules [5].

In many real implementations, adopting "micro digital twins" (small, scoped twins for machines/cells/segments) is a pragmatic approach: instead of attempting an enterprise-wide monolithic twin, organizations deploy smaller twins that can later be composed. Methodological contributions for developing digital twins in industry often emphasize structured development steps, a communication layer, and clear interfaces to operational data sources and decision modules. Cloud-based digital twin implementations demonstrate that remote monitoring and control can be realized when connectivity, data management, and application logic are aligned (including MQTT in the communication layer) [6,7].

2.3 Discrete Event Simulation as an experimentation substrate

Discrete Event Simulation (DES) is one of the most widely used simulation approaches in production system modeling and remains a dominant trend in recent reviews. It is frequently used to represent process flows, resource constraints, and stochastic behaviors while enabling KPI-focused experimentation (throughput, lead time, utilization). Evidence from literature reviews and software adoption studies indicates that Plant Simulation is a common DES tool in smart manufacturing research and application [2,8].

Beyond standalone use, DES becomes especially valuable when coupled with data-driven methods. Studies show DES can generate structured datasets for analytics/Machine Learning (ML), while ML can enrich simulation-based decision support (e.g., planning for flexible job shops). This line of work supports a "simulation + learning" loop, aligned with digital twin objectives when the simulation is connected to evolving system data [4].

2.4 Event-driven architecture and interoperability (MQTT, OPC UA)

Event-driven architecture is frequently proposed to improve responsiveness and decouple systems in industrial environments. Manufacturing-oriented event-driven approaches have been explored through architectures that model production information flows as events and support scalable integration of line-level systems. In industrial IoT practice, MQTT provides lightweight publish/subscribe messaging that naturally supports event-driven interaction across distributed components, while OPC UA provides structured information models and industrial-grade connectivity patterns [9].

Protocol selection is not neutral. Survey work on IoT protocols highlights performance and suitability trade-offs across application contexts, while comparative analyses continue to examine MQTT alongside alternatives such as CoAP [10,11]. Conceptual modeling research on MQTT for IoT-based systems further supports systematic understanding of messaging patterns and their role in system design [12].

For OPC UA, interoperability research includes mechanisms for legacy integration (e.g., OPC-UA agents for legacy PLCs), and implementations addressing communication control in embedded OPC UA servers [13,14]. OPC UA also appears in digital-twin-oriented information modeling work (e.g., OPC UA–based workshop information models). Official reference architecture materials provide a consolidated view of OPC UA concepts and how they map to system integration needs [15,16].

Finally, security and robustness must be considered for event-driven integration. Studies reviewing IoT protocol security features emphasize that protocol-level choices and deployment practices significantly affect risk exposure. Intrusion detection approaches tailored for MQTT environments further indicate the need to treat IIoT messaging as a security-critical subsystem [17,18].

2.5 Cognitive technologies: from analytics to decision logic (agents, hybrid intelligence)

Within manufacturing, "cognitive" capabilities are commonly associated with perception (state awareness), reasoning (evaluation of alternatives), learning (data-driven improvement), and action (recommendations or automated control). Systematic reviews of ML in production lines categorize tasks such as prediction, classification, anomaly detection, and optimization support, providing a foundation for selecting minimal algorithms suitable for prototype-level decision logic. Predictive quality reviews further indicate that ML/DL (Deep Learning) methods are increasingly used to support quality-related decisions, though integration into operational workflows remains challenging [19,20].

A practical "cognitive" step beyond analytics is using agent logic (rule-based or ML-based) to transform events into decisions. This aligns with hybrid intelligence perspectives where ML supports—but does not necessarily replace—human decision-making in production management. Recent systematic work on hybrid intelligence highlights decision-support gaps and opportunities to structure human-AI collaboration across production decisions [21].

3 Methodology

3.1 Research approach

This work follows a prototype-driven research approach aimed at demonstrating core integration principles relevant to cognitive enhancements of digitally supported manufacturing systems. Instead of claiming a full digital twin, the implemented artifact is positioned as a simulation-based micro-prototype (an event-driven simulation demonstrator). The objective is to show, in a transparent and reproducible manner, how a discrete-event simulation model can expose process-relevant state transitions as messages and be influenced by external decision logic via broker-mediated commands.

The methodological focus is therefore on observability, decoupled communication, and closed-loop actuation rather than on industrial-scale optimization, calibration to physical assets, or comprehensive validation.

3.2 Overall architecture

The demonstrator is organized as four loosely coupled layers:

1) Simulation layer (Tecnomatix Plant Simulation): a minimal discrete-event model representing a simple production flow and SimTalk methods responsible for publishing messages and processing external commands.
2) Messaging layer (MQTT broker): an event-driven communication backbone enabling publish/subscribe interaction between the simulation and external components.
3) External interaction layer:
   - a mobile MQTT client (MQTT X) for manual monitoring and command testing,
   - a lightweight Python MQTT bridge used for structured logging and reproducible experiments (and optionally as a connection point for future automated agent logic).
4) Evidence layer: screenshots from Plant Simulation and MQTT X, plus log artifacts generated by the Python bridge (bridge.jsonl, telemetry.csv) supporting traceability and replication.

This separation keeps the simulation model readable and minimal while allowing external decision logic to evolve from manual commands to automated policies without redesigning the core simulation.

3.3 Simulation micro-prototype in Tecnomatix Plant Simulation

A minimal discrete-event model was implemented in Tecnomatix Plant Simulation consisting of four core objects: Source (entity generation), Station (single processing resource), Drain (sink), and an EventController managing simulation execution. The simulation micro-prototype is shown in Figure 1. The processing behavior is intentionally simple to ensure that integration mechanisms remain transparent and interpretable.

The Station object uses two standard discrete-event triggers as integration points:

- OnEntry: executed when an entity enters the Station (processing start),
- OnExit: executed when an entity leaves the Station (processing completion).

Both triggers call custom SimTalk methods that publish messages to the MQTT broker. In addition, the model exposes a controllable processing parameter (Station.ProcTime) that can be updated externally to demonstrate command-driven actuation.

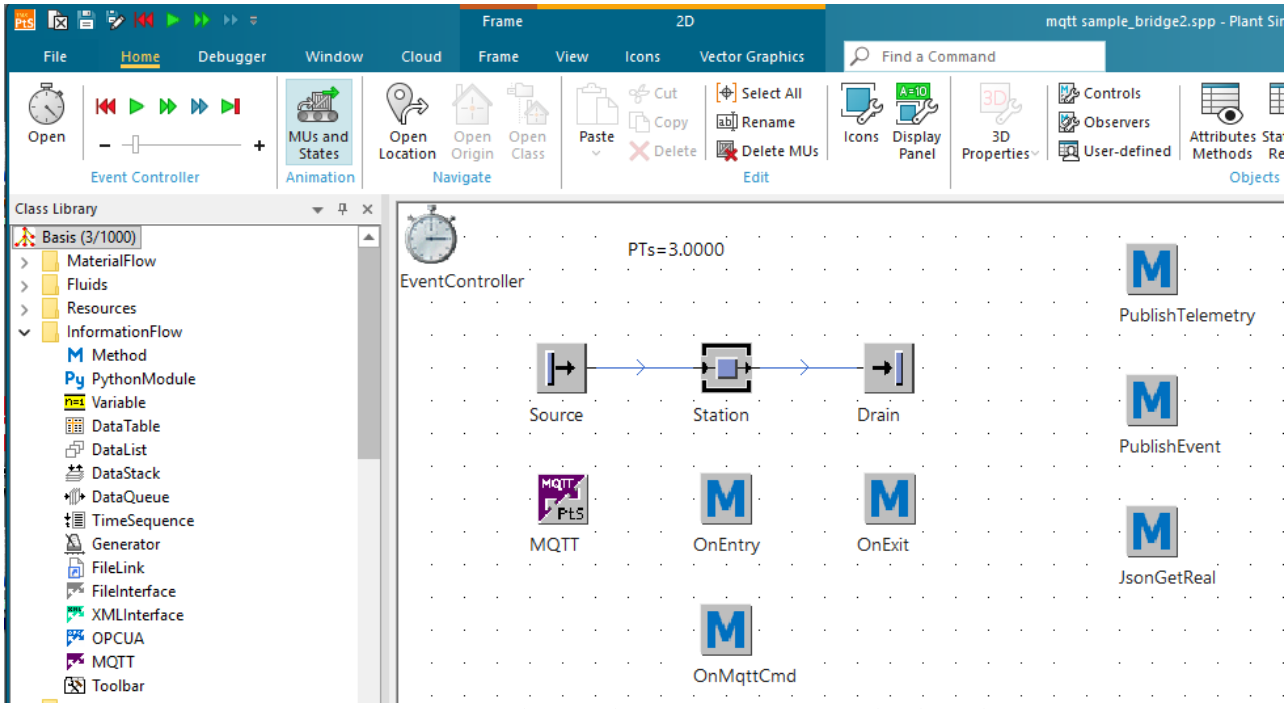

Figure 1 Tecnomatix Plant Simulation micro-prototype used in the study (Source–Station–Drain with event-driven integration elements)

3.4 MQTT connectivity inside Plant Simulation

Communication between simulation and external components is enabled through an MQTT connector object configured with the broker host and TCP port (Figure 2). The connector uses a callback method (e.g., OnMqttCmd) to process incoming messages, while outgoing messages are produced by dedicated publishing helpers.

Two categories of SimTalk methods support modular integration:

- Publishing helpers (e.g., PublishEvent, PublishTelemetry) responsible for formatting and publishing outgoing messages,
- Command callback (e.g., OnMqttCmd) responsible for parsing and validating incoming commands and applying corresponding actions in the simulation.

This separation ensures that event/telemetry publishing is decoupled from command processing and that both remain separate from the core simulation objects.

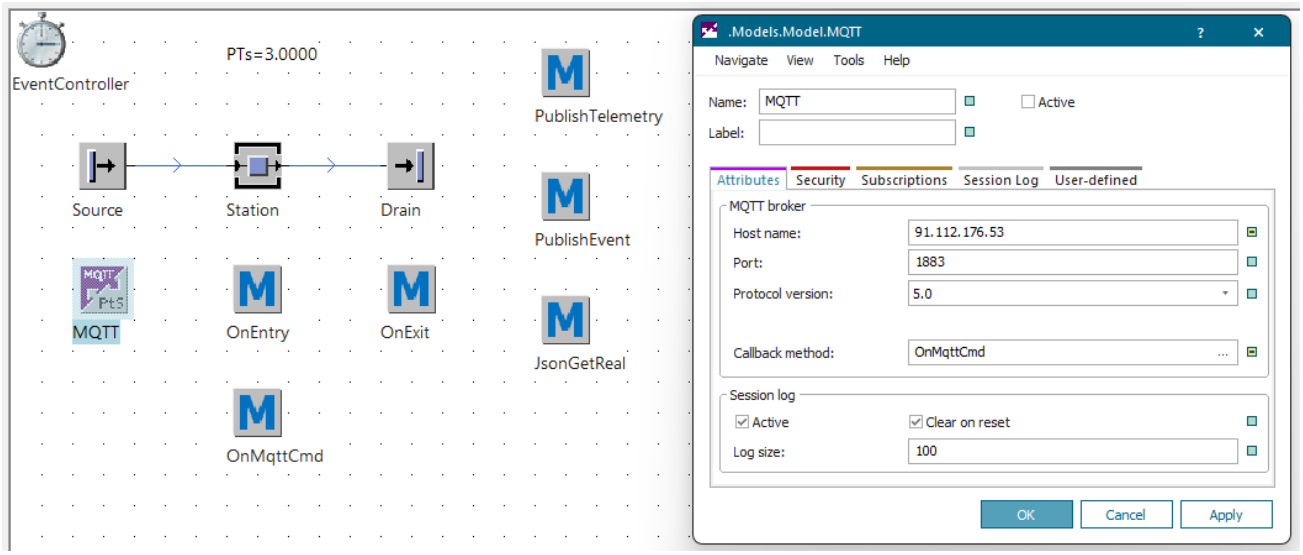

Figure 2 MQTT connector settings in Plant Simulation (broker parameters and callback method for incoming command)

3.5 Message design and topic conventions

During development, two topic styles were used, serving complementary purposes:

- Human-readable manual testing namespace (MQTT X):
  plantsim/station, plantsim/cmd, plantsim/alert
  This namespace supports rapid manual testing and visibility of message exchange using a mobile MQTT client.
- Structured experimental namespace (Python bridge + logging):
  sim/plant1/telemetry/<tag>, sim/plant1/events/<name>, sim/plant1/cmd/<command>, sim/plant1/ack/<command>
  This namespace supports systematic logging, traceability, and repeatable evaluation.

Although the namespaces differ, the functional meaning remains consistent. The simulation emits events/telemetry ("perception") and external components publish commands ("action"), forming a minimal closed loop.

The primary perception signals in the demonstrator are discrete events representing meaningful state transitions of the Station. Specifically, OnEntry publishes a "processing start" message and OnExit publishes a corresponding "processing finish" message. For interpretability during testing, messages include simulation time references and a wall-clock timestamp, enabling reconstruction of the event sequence and approximate cycle-time behavior at the message level (figure 3).

Incoming commands are used for two actuation categories:

- Execution control: START and STOP commands trigger the EventController to start or stop simulation execution.
- Parameter update: a compact command format PT:<value> updates the Station processing time (Station.ProcTime), providing a clear actuation handle for external decision logic.

Command parsing includes minimal validation to ensure robust behavior during manual testing and scripted experiments.

```
param topic : string, payload : string
  var s    : string := strTrim(payload);
  var snum : string;
  var val  : real;
  print "CMD: " + topic + " :: " + payload;
  if (s = "START") or (s = "start") then
    eventController.start;
    root.MQTT.publish("sim/plant1/ack/start", "{ \"ok\": true }", false);
  elseif (s = "STOP") or (s = "stop") then
    eventController.stop;
    root.MQTT.publish("sim/plant1/ack/stop", "{ \"ok\": true }", false);
  else
    snum := s;
    if (strLen(s) >= 2) and (strCopy(s,1,1) = "\"") and (strCopy(s, strLen(s), 1) = "\"") then
      snum := strCopy(s, 2, strLen(s)-2);
    end;
    val := str_to_num(snum);
    if (val = 0) and (find(s, "{") = 1) then
      val := root.JsonGetReal(s, "value");
    end;
    print "parsed value = " + to_str(val, 6);
    if val > 0 then
      root.Station.ProcTime := val;
      root.MQTT.publish("sim/plant1/ack/set_pt", "{ \"ok\": true, \"applied\": " + to_str(val, 6) + " }", false);
    else
      root.MQTT.publish("sim/plant1/ack/set_pt", "{ \"ok\": false, \"error\": \"invalid number\" }", false);
    end;
  end;
```

.Models.Model .Models.Model.OnMqttCmd

Figure 3 Excerpt of the SimTalk callback implementing command handling (START/STOP) and processing-time update (PT)

To prevent unrealistic parameter settings during testing, the demonstrator implements a basic constraint check for processing time. If the received value violates predefined limits, the system publishes an alert message to a dedicated alert topic (e.g., plantsim/alert). This separation of operational messages (plantsim/station) from anomaly signaling (plantsim/alert) increases transparency and provides a minimal guardrail mechanism suitable for repeatable experiments.

3.6 Python-based MQTT bridge for structured logging and reproducibility

To complement manual MQTT testing, a lightweight Python MQTT bridge was implemented. The bridge subscribes to all topics under a configurable base namespace (BASE_TOPIC, default sim/plant1/#) and performs:

- message ingestion (topic + payload capture with decoding and optional JSON parsing),
- channel-based routing (classification by topic structure into telemetry, events, and commands),
- persistent logging (telemetry export to CSV and full message stream to JSONL),
- acknowledgement publishing (ack/*) to confirm broker-mediated command delivery.

Two log files are produced:

- bridge.jsonl (JSON Lines): append-only message stream preserving topic context and raw payload for traceability and debugging,
- telemetry.csv: structured telemetry table designed for offline analysis (e.g., Excel), containing fields such as ts, run_id, topic, tag, value, unit, and raw_json.

The exported telemetry evidence is evaluated and discussed in Section 4 Results.

A small publisher script was used to generate repeatable telemetry sequences (e.g., conveyor speed values) and send example command payloads, enabling validation of broker connectivity, payload parsing, and logging correctness independently of simulation runtime. This improves reproducibility and supports controlled testing of message structures.

3.7 Independent verification using MQTT X (mobile client)

Message exchange was independently verified using MQTT X on a smartphone (Figure 4). The client was used to:

- subscribe to station/event topics and observe start/finish messages,
- publish commands (START/STOP, PT updates),
- observe alerts triggered by out-of-range processing time updates.

This verification provides evidence that the demonstrator is observable and controllable using standard MQTT tooling and reinforces the claim of broker-mediated decoupling between simulation and external decision logic.

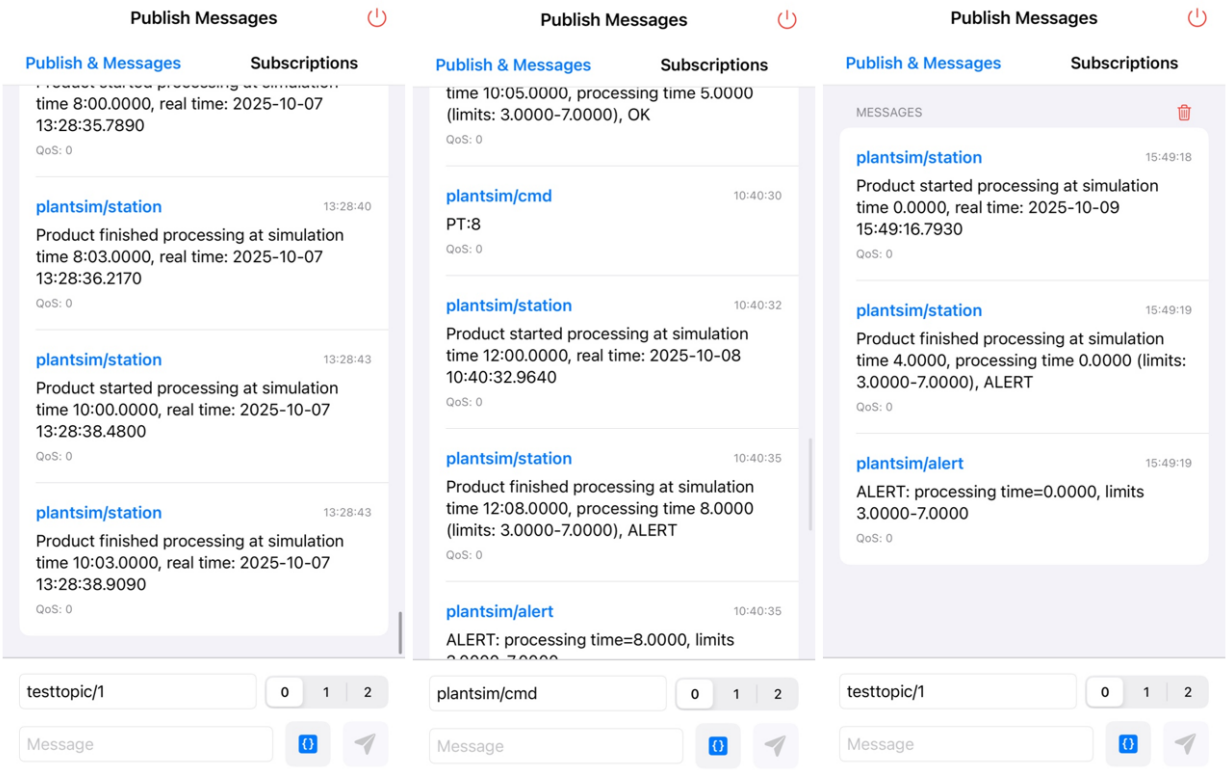

Figure 4 MQTT X mobile client verification of broker-mediated interaction (station messages, commands, and alert signalling)

3.8 Experimental procedure and collected evidence

The demonstrator feasibility was evaluated through repeated runs using the following procedure:

1) start simulation either locally or via command topic,
2) observe event stream: station start/finish events published on the station/event topic(s),
3) send commands (START/STOP, PT updates) via MQTT X or publisher script,
4) verify actuation through subsequent event timing/behavior changes,
5) intentionally trigger a constraint violation and verify alert publication,
6) captured evidence: Plant Simulation screenshots (model layout, trigger configuration, MQTT connector settings, key SimTalk methods), MQTT X screenshots (topics and payloads), and structured logs (bridge.jsonl, telemetry.csv) generated by the Python bridge.

This evidence package supports transparency and enables replication of the demonstrated event-driven integration workflow.

Table 1 Topic patterns and message types used in the demonstrator (paste-ready)

Topic pattern	Direction	Purpose	Example payload
plantsim/station	Simulation → Broker → MQTT X	Manual event monitoring (start/finish)	Text message with simulation/wall time
plantsim/cmd	MQTT X → Broker → Simulation	Manual commands (START/STOP, PT:<v>)	START, STOP, PT:5
plantsim/alert	Simulation → Broker → MQTT X	Alerts for violated limits	"ALERT: processing time=…, limits …"
sim/plant1/telemetry/<tag>	Publisher/Simulation → Broker → Bridge	Structured telemetry logging	{"ts":"…","run_id":"…", "value":1.2, "unit":"m/s"}
sim/plant1/events/<name>	Simulation → Broker → Bridge	Structured events	{"ts":"…", "severity": "info", "msg":"…"}
sim/plant1/cmd/<command>	Client → Broker → Bridge/Simulation	Structured commands	JSON or text command payload
sim/plant1/ack/<command>	Bridge → Broker → Client	Delivery acknowledgement	{"ok":true, "ts":"…", "echo":…}

4 Results and discussion

4.1 Broker-mediated observability of simulation behavior

The first result is the successful exposure of simulation behavior through broker-mediated messaging. The Plant Simulation micro-prototype continuously published station-level process events (start and finish) which were observable using an independent MQTT client (MQTT X, see Fig. 4). This demonstrates that meaningful discrete-event transitions can be externalized as an event stream without direct access to internal simulation objects.

From an integration perspective, this improves observability: external components can reconstruct the sequence of operations and infer basic cycle-time behavior from "start/finish" event pairs. Even in a minimal single-station setup, the message stream represents a practical foundation for monitoring and for triggering downstream logic (e.g., threshold checks, notifications, or simple control policies).

4.2 Command-driven actuation and closed-loop interaction

A second key result is the validated closed-loop interaction between external clients and the simulation model. Commands issued via MQTT affected simulation execution and parameters through the broker, confirming that the simulation can be influenced through a decoupled control channel.

Two actuation categories were demonstrated:

- Execution control: START/STOP commands triggered the simulation execution controller, which was reflected by the presence/absence of subsequent station event messages.
- Parameter update: processing-time update commands in the format PT:<value> modified the station processing time (Station.ProcTime). The effects were indirectly observable through the station event stream and through subsequent behavior during repeated runs.

This result is significant for cognitive integration, while the demonstrator does not claim autonomous optimization, it establishes a clean interface where external decision logic can be attached and iterated independently from the simulation model.

4.3 Constraint handling and alert signaling

To improve robustness and transparency of external actuation, a basic constraint mechanism for processing time was introduced. When a received processing-time value violated predefined bounds, an alert message was published on a separate topic (plantsim/alert). This separation between operational event reporting (station start/finish) and anomaly signaling (alert) supports clearer interpretation, easier debugging, and safer experimentation.

From a design standpoint, even a minimal guardrail mechanism is useful because it formalizes the boundary between "allowed actions" and "invalid actions" and provides feedback to external components in a machine-observable form. This is particularly relevant once an automated agent replaces manual commands.

4.4 Structured logging and evidence for reproducibility

Beyond message-level verification, the Python MQTT bridge produced structured logs that enable reproducible inspection and offline analysis. The bridge captured incoming messages under the structured namespace (sim/plant1/...), stored a complete audit stream in bridge.jsonl, and exported telemetry data into telemetry.csv.

The exported telemetry table includes timestamp (ts), experiment identifier (run_id), topic and derived tag, numeric value, unit, and the raw JSON payload. This enables immediate inspection in spreadsheet tools and supports downstream processing for analysis pipelines. An example excerpt of the telemetry export opened in Excel is shown in Figure 5.

This result strengthens the demonstrator's value as an experimental platform, it does not only "send messages," but also produces structured datasets that can be used to validate message schemas, compare runs, and later evaluate decision policies.

4.5 Discussion: implications for cognitive, event-driven integration

The demonstrated results support three integration implications:

1) Decoupling is feasible at small scale and scales conceptually.
   The simulation remains focused on process behaviour, while external logic interacts through topics and payload conventions. This reduces coupling and allows independent evolution of external components (manual client → rule-based agent → advanced AI component).
2) Events provide a natural perception layer for cognitive logic.
   Discrete-event transitions (start/finish) are interpretable, stable integration points. Even without deep state exposure, event streams can trigger simple reasoning and decision logic.
3) Command topics provide an explicit actuation layer.

The PT:<value> command illustrates a minimal but clear actuation handle. While simplistic, it proves that actions can be applied externally and verified through subsequent event behaviour and acknowledgements/alerts.

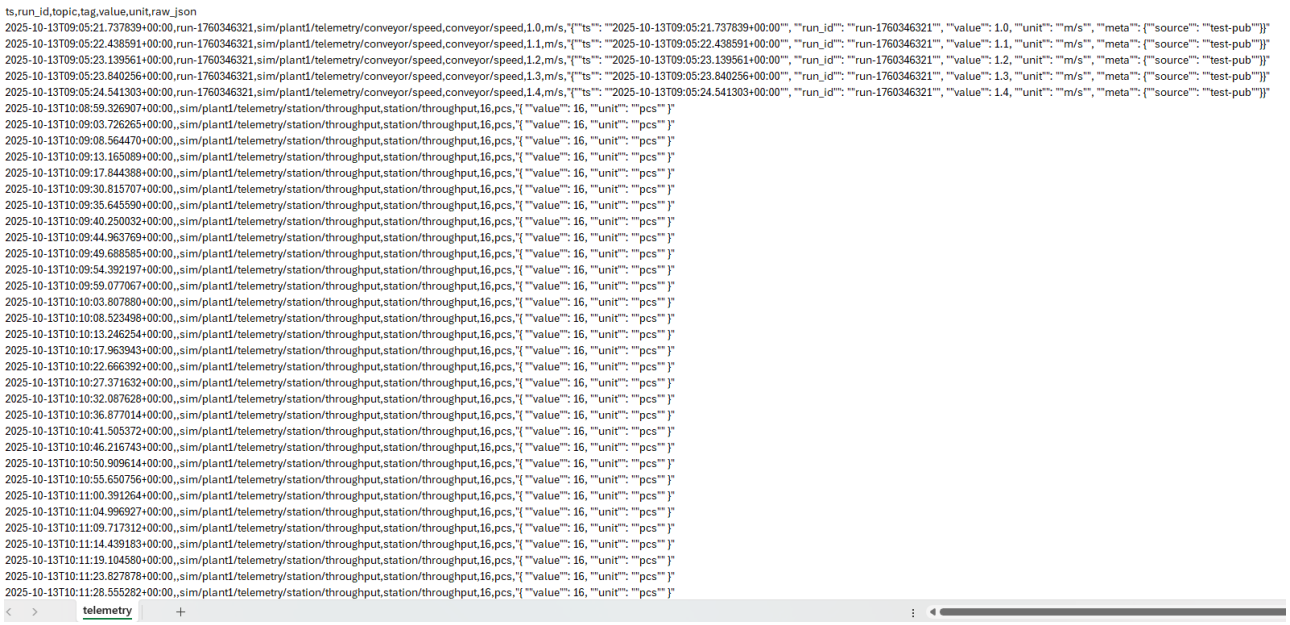

Figure 5 Example of exported telemetry log (telemetry.csv) generated by the Python bridge for offline analysis (ts, run_id, topic/tag, value, unit, raw payload)

In digital value chain terms, the demonstrator approximates a minimal perception–decision–action loop using widely available tooling (DES + MQTT + Python). It therefore provides a concrete base for further research on how cognitive modules can be systematically integrated into digitally supported manufacturing workflows.

4.6 Limitations and future work

The presented demonstrator is intentionally minimal and has several limitations:

- Scale and complexity: the model represents a single processing station and does not cover multi-resource coordination, routing decisions, or variability across multiple processes.
- Message semantics: the manual testing namespace uses human-readable text messages, future iterations should adopt fully structured schemas (e.g., JSON for all channels) with consistent typing and versioning.
- Validation scope: the demonstrator is a simulation-based experimental setup and is not calibrated to a physical production asset. Therefore, results should be interpreted as architectural feasibility rather than operational performance claims.
- Decision logic: cognitive behaviour is represented by external command capability rather than by a validated autonomous policy. Future work should be integrated into a rule-based or learning-based agent and evaluate decisions against defined KPIs.

Next steps include extending the model to multi-station flows, enriching telemetry (queue lengths, utilization, throughput), formalizing message schemas, adding acknowledgements consistently, and integrating automated decision policies that can be evaluated using the bridge-generated datasets.

5 Conclusions

This paper presented a simulation-based micro-prototype that demonstrates how event-driven communication can be used to integrate discrete-event simulation with external cognitive decision logic in a transparent and reproducible manner. Instead of proposing comprehensive digital twin architecture, the study deliberately focused on a minimal and modular setup that enables closed-loop interaction between perception (event streams), decision-making (external logic), and action (broker-mediated commands).

The main contribution of this work is twofold. First, it proposes a conceptual framework that links digital value chain integration with cognitive technologies through event-driven principles. Second, it validates this framework through a small-scale but functional prototype implemented in Tecnomatix Plant Simulation and connected via MQTT to external monitoring, control, and logging components. The results confirm that meaningful simulation events can be externalized, acted upon, and systematically logged without embedding complex logic directly into the simulation model.

The results showed that simulation behavior can be externalized as a broker-mediated event stream (station start/finish messages) and that external commands can influence both execution and model parameters (START/STOP and

processing-time updates). A basic constraint mechanism with alert signaling improved robustness and interpretability during testing. In addition, a Python MQTT bridge enabled structured logging (JSONL/CSV), strengthening traceability and supporting offline inspection of telemetry data.

Although the demonstrator is intentionally limited in scale and complexity, it provides a reproducible experimental foundation for future research. The proposed approach can be extended toward multi-station systems, richer telemetry, structured message schemas, and automated decision policies. As such, the presented micro-prototype represents a practical stepping stone toward more advanced cognitive and adaptive digital value chain implementations.

Acknowledgement
This work was supported through the Research Mobility Call, an activity financed by the EIT Manufacturing – Slovakia–X Fund, a joint initiative of EIT Manufacturing and the Ministry of Investment, Regional Development, and Informatization of the Slovak Republic. The prototype and experimental evidence presented in this paper were developed during research stay at MCI | The Entrepreneurial School® in Innsbruck, Austria and by the project VEGA 1/0210/25 and KEGA 014TUKE-4/2025 granted by the Ministry of Education, Research, Development and Youth of the Slovak Republic.

References

[1] WU, K., ZHENG, M.: Industry 4.0: review and proposal for implementing a smart factory, *The International Journal of Advanced Manufacturing Technology*, Vol. 133, pp. 1331-1347, 2024. https://doi.org/10.1007/s00170-024-13839-7

[2] CIAUŞU-SLIWA, M.R., CIAUŞU-SLIWA, D., DODUN, O.: Discrete-Event Simulation for Smart Manufacturing: Assessing Software, Industry Adoption, and Future Research Challenges, *Bulletin of the Polytechnic Institute of Iaşi. Machine constructions Section*, Vol. 71, No. 1, pp. 105-124, 2025. https://doi.org/10.2478/bipcm-2025-0009

[3] MUSAIGWA, M., SWANEPOEL, J.A.: The Impact of Digital Transformation on Supply Chain Optimisation in Industrial Engineering: A Systematic Literature Review, *International Journal of Applied Research in Business and Management*, Vol. 6, No. 1, pp. 1-23, 2025. https://doi.org/10.51137/wrp.ijarbm.2025.mmtt.45812

[4] ELBASHEER, M., LONGO, F., MADDEN, M.G., PADOVANO, A., ULLAH, I.: Towards a Digital Twin for Production Planning: Combining Discrete Event Simulation and ML for Flexible Job Shops, *Procedia Computer Science*, Vol. 253, pp. 3078-3087, 2025. https://doi.org/10.1016/j.procs.2025.02.032

[5] REDELINGHUYS, A.J.H., BASSON, A.H., KRUGER, K.: A six-layer architecture for the digital twin: a manufacturing case study implementation, *Journal of Intelligent Manufacturing*, Vol. 31, pp. 1383-1402, 2020. https://doi.org/10.1007/s10845-019-01516-6

[6] GONZÁLEZ-HERBÓN, R., GONZÁLEZ-MATEOS, G., RODRÍGUEZ-OSSORIO, J.R., DOMÍNGUEZ, M., ALONSO, S., FUERTES, J.J.: An Approach to Develop Digital Twins in Industry, *Sensors*, Vol. 24, No. 3, pp. 1-15, 2024. https://doi.org/10.3390/s24030998

[7] BIN TOUHID, M.T., MARNE, M., OSKROBA, T., MIRAHMADI, S.A., ZHU, E., MEHRABIAN, A., DEFERSHA, F., YANG, S.: Building a cloud-based digital twin for remote monitoring and control of a robotic assembly system, *The International Journal of Advanced Manufacturing Technology*, Vol. 129, No. 9-10, pp. 4045-4057, 2023. https://doi.org/10.1007/s00170-023-12611-7

[8] KURNIASIH, J., ABAS, Z.A., ASMAI, S.A., WIBOWO, A.B.: Process Mining and Simulation Modeling in Production Systems: A Review, *Logforum*, Vol. 20, No. 4, pp. 429-453, 2024. https://doi.org/10.17270/J.LOG.001092

[9] GROEN, D., MULATIER, C.D., PASZYNSKI, M., KRZHIZHANOVSKAYA, V.V., DONGARRA, J.J., SLOOT, P.M.A.: *Computational Science – ICCS 2022*, 22nd International Conference, London, UK, June 21-23, 2022, Proceedings, Part III, in Lecture Notes in Computer Science, Vol. 13352, Cham: Springer International Publishing, 2022. https://doi.org/10.1007/978-3-031-08757-8

[10] BAYILMIŞ, C., EBLEME, M.A., ÇAVUŞOĞLU, Ü., KÜÇÜK, K., SEVIN, A.: A survey on communication protocols and performance evaluations for Internet of Things, *Digital Communications and Networks*, Vol. 8, No. 6, pp. 1094-1104, 2022. https://doi.org/10.1016/j.dcan.2022.03.013

[11] KASHYAP, M., SHARMA, V.: A comparative analysis and implementation of CoAP and MQTT protocol for IoT communication, *Life Cycle Reliability and Safety Engineering*, Vol. 14, pp. 715-730, 2025. https://doi.org/10.1007/s41872-025-00324-7

[12] MOHAMMAD EL-BASIONI, B.M.: A conceptual modelling approach of MQTT for IoT-based systems, *Journal of Electrical Systems and Information Technology*, Vol. 11, pp. 1-31, 2024. https://doi.org/10.1186/s43067-024-00181-x

[13] LEE, S.Y., SUNG, M.: OPC-UA Agent for Legacy Programmable Logic Controllers, *Applied Sciences*, Vol. 12, No. 17, pp. 1-20, 2022. https://doi.org/10.3390/app12178859

[14] NGUYEN, N.T., PIMENIDIS, E., KHAN, Z., TRAWINSKI, B.: *Computational Collective Intelligence*, Springer International Publishing, 2018. https://doi.org/10.1007/978-3-319-98443-8

[15] FREITAS, L.: OPC-UA in interoperability - a performance comparative testing, *IFAC-PapersOnLine*, Vol. 58, No. 8, pp. 240-245, 2024. https://doi.org/10.1016/j.ifacol.2024.08.127
[16] WU, M., YING, W.: Research and application of OPC UA-based digital twin smoke alarm calibration workshop information model, *PLoS One*, Vol. 19, No. 7, pp. 1-18, 2024. https://doi.org/10.1371/journal.pone.0307801
[17] RIZZARDI, A., SICARI, S., COEN-PORISINI, A.: Analysis on functionalities and security features of Internet of Things related protocols, *Wireless Networks*, Vol. 28, pp. 2857-2887, 2022. https://doi.org/10.1007/s11276-022-02999-7
[18] ZEGHIDA, H., BOULAICHE, M., CHIKH, R., PATEL, A., BARROS, A.L.B., BAMHDI, A.M.: XMID-MQTT: explaining machine learning-based intrusion detection system for MQTT protocol in IoT environment, *International Journal of Information Security*, Vol. 24, pp. 1-22, 2025. https://doi.org/10.1007/s10207-025-01036-w
[19] TERCAN, H., MEISEN, T.: Machine learning and deep learning based predictive quality in manufacturing: a systematic review, *Journal of Intelligent Manufacturing*, Vol. 33, pp. 1879-1905, 2022. https://doi.org/10.1007/s10845-022-01963-8
[20] KANG, Z., CATAL, C., TEKINERDOGAN, B.: Machine learning applications in production lines: A systematic literature review, *Computers & Industrial Engineering*, Vol. 149, pp. 1-11, 2020. https://doi.org/10.1016/j.cie.2020.106773
[21] SAUER, C.R., BURGGRÄF, P., STEINBERG, F.: A systematic review of machine learning for hybrid intelligence in production management, *Decision Analytics Journal*, Vol. 15, pp. 1-15, 2025. https://doi.org/10.1016/j.dajour.2025.100574

Review process
Single-blind peer review process.