

PROGRAMMING AND GRAPHICAL REFINEMENT OF TECNOMATIX PLANT SIMULATION MODELS

PÉTER KOVÁTS¹–RÓBERT SKAPINIECZ²–PÉTER TAMÁS³–GORAN DUKIC⁴

Abstract: Large discrete-event simulation models must represent the analysed logistics system in sufficient detail while remaining stable, transparent and computationally manageable. This paper presents a practice-oriented framework of programming and graphical refinement principles through a Tecnomatix Plant Simulation case study of a self-designed intermodal container terminal. In this paper, refinement denotes development-oriented improvement of the executable model and is distinguished from mathematical simulation optimization or a demonstrated runtime speed-up. The methodological contribution integrates indexed generalization of repeated control logic, explicit reserve-release state management, decoupled process completion rules, reproducible initialization and configurable 3D fidelity within one development framework. The final model contains nine loading areas with 16 pickup points per area for 20-foot containers (144 in total), corresponding to eight paired pickup positions per area for 40-foot containers (72 in total), nine storage blocks, nine RTG-type cranes and 3,456 twenty-foot container positions. The evaluation therefore uses structural evidence and development criteria - maintainability, extensibility, stability, repeatability and graphical efficiency - while computational benefits are stated cautiously because versioned hardware-independent runtime, memory and initialization benchmarks were not recorded. The resulting framework provides transferable guidance for large object-based simulation models and defines quantitative indicators for future controlled benchmarking.

Keywords: discrete-event simulation, Plant Simulation, model development, model maintainability, 3D visualization, intermodal terminal

1. INTRODUCTION

Discrete-event simulation is widely used to analyse logistics systems whose behaviour is determined by interacting resources, queues, transport operations and control rules. It allows alternative layouts and operating policies to be tested before physical implementation, and it also supports bottleneck analysis, capacity planning and the evaluation of scheduling decisions [1-8]. In intermodal terminals, the simultaneous movement of trucks, containers and handling equipment creates a particularly complex modelling task because material flow and traffic flow are strongly coupled. A local change in gate capacity, routing logic or crane assignment can alter waiting times and congestion in other parts of the system. Therefore, a simulation model must not only reproduce the logistics process correctly; it must also be designed so that it can be modified, debugged and executed reliably.

As a model grows, programming structure and graphical representation become increasingly important. Repeated object-specific code increases the probability of inconsistent modifications and hidden state errors, while excessive synchronization can

¹ PhD student, Institute of Logistics, University of Miskolc, Hungary
peter.kovats@uni-miskolc.hu

² Associate professor, Institute of Logistics, University of Miskolc, Hungary
robert.skapinyecz@uni-miskolc.hu

³ Full Professor, Institute of Logistics, University of Miskolc, Hungary
peter.tamas@uni-miskolc.hu

⁴ Full Professor, Faculty of Mechanical Engineering and Naval Architecture, University of Zagreb, Croatia
goran.dukic@fsb.unizg.hr

create artificial deadlocks. The graphical layer creates a similar trade-off: detailed 3D objects support verification and presentation, but imported CAD components and numerous repeated moving objects can increase file size and rendering demand. A large model must therefore balance process detail, software transparency and visual complexity.

This paper presents a structured, practice-oriented approach to programming and graphical refinement using a self-designed intermodal container terminal model. The contribution is methodological rather than algorithmic: the work does not propose a new terminal operating policy or a new mathematical optimization method. Instead, it treats the executable simulation model itself as the development target and combines generic control logic, explicit state management, controlled synchronization, reproducible initialization and purpose-dependent graphical fidelity in a single Plant Simulation framework. The final model contains nine storage blocks, nine cranes, 144 pickup points for 20-foot containers, equivalent to 72 paired pickup positions for 40-foot containers, and 3,456 twenty-foot container positions. The article therefore focuses on structural and methodological consequences and does not claim a hardware-specific numerical speed-up.

2. RESEARCH CONTEXT AND OBJECTIVE

2.1. Model performance beyond simulation results

The performance of a simulation model has two distinct meanings. Operational performance is expressed through throughput, waiting time and resource utilization, whereas model performance concerns stability, readability, modifiability, memory and graphical demand, and repeatable initialization. A model can produce meaningful logistics indicators while containing fragile control logic, or it can use impressive graphics while becoming unnecessarily difficult to execute and maintain.

Container-terminal simulation studies generally focus on capacity, congestion, equipment coordination and storage policies [1-6, 9-11], while classical simulation-optimization research primarily addresses the systematic selection of model inputs or alternatives based on output performance [7, 8]. A different but complementary strand of simulation-modelling research emphasizes model continuity, abstraction, modularity, separation of concerns and maintainability during the development of executable simulation models [12, 13]. Verification and validation literature similarly stresses structured development, staged checking and documented model behaviour [14]. For the graphical layer, empirical studies show that 3D visualization can support verification, understanding and communication, while excessive fidelity can add modelling and computational cost; the appropriate level of detail should therefore be linked to the purpose of the model [15, 16]. The practical gap addressed in this paper is the translation of these general recommendations into a coherent set of Plant Simulation-specific programming and graphical development principles for a large logistics model.

2.2. Research objective and evaluation criteria

The objective is to systematize the development measures applied during model construction and to evaluate them using five criteria: maintainability, extensibility, stability, repeatability and graphical efficiency. Throughout the paper, refinement denotes the removal of avoidable structural, synchronization and graphical complexity. It is used as a development-oriented

concept and does not imply mathematical optimization of operating parameters or a proven reduction in wall-clock execution time. The evaluation is based on the structure and behaviour observed during model development. Controlled before-and-after runtime, peak-memory and initialization-time measurements were not preserved; consequently, such effects are identified as expected or proposed for future benchmarking rather than presented as demonstrated performance gains.

2.3. Scientific novelty and methodological contribution

The scientific novelty lies in integrating several development-level decisions that are often treated separately into one explicit framework for large Plant Simulation models. The framework links five coupled principles - indexed generalization of repeated object logic, paired reserve-release state transitions, local rather than circular synchronization, reproducible initialization and purpose-dependent graphical fidelity - to the five evaluation criteria defined above. Its contribution is therefore a reusable modelling method and decision structure rather than a new discrete-event simulation algorithm. The intermodal terminal case demonstrates the framework at a non-trivial scale: nine loading areas with 144 pickup points for 20-foot containers (or 72 paired positions for 40-foot containers) are handled through generalized addressing, nine crane resources use consistent state handling, and 3,456 storage positions are initialized and observed within the same model architecture. This positions the proposed approach as complementary to operational simulation optimization: the latter searches for better system configurations, whereas the present framework aims to make the model used for those experiments more transparent, repeatable and scalable.

3. CASE STUDY MODEL

3.1. Architecture and scale

The case study is a generalized intermodal container terminal developed in Tecnomatix Plant Simulation [17]. It does not reproduce the confidential operation of a specific company; instead, it combines typical terminal processes in a form suitable for controlled experiments. Trucks and their associated containers are generated on the right-hand side of the model. Each truck carries two twenty-foot containers. After generation, vehicles pass an OCR identification gate and a checkpoint where the model determines the assigned loading area and storage block. Trucks then travel on the internal road network to one of nine loading areas. Each loading area is connected to one storage block and one RTG-type crane. After unloading, trucks leave through a separate exit route so that outbound traffic interferes as little as possible with arriving and waiting vehicles.

Each storage block contains 16 longitudinal, 6 transverse and 4 vertical positions. The total theoretical storage capacity is therefore $9 \times 16 \times 6 \times 4 = 3,456$ twenty-foot container positions. Each loading area contains 16 pickup points for 20-foot containers, giving $9 \times 16 = 144$ such points in the complete terminal. For 40-foot containers, the points are used in pairs, resulting in eight 40-foot pickup positions per loading area and $9 \times 8 = 72$ positions in total. The multiple pickup-point design reduces unnecessary loaded crane travel and permits a more balanced distribution of handling operations. The model also includes checkpoint and checkout states, truck counters, time logs, crane task lists, storage coordinates and termination conditions (Table I). The full system is illustrated in Fig. 1.

Table I.
Main elements and scale of the case study model

Model element	Quantity or function
Truck and container generators	Create arriving vehicles and two 20-foot containers per truck
OCR gate	Vehicle and container identification
Checkpoint / checkout resources	Entry and exit control with explicit occupancy states
Loading areas	9 separate truck-service areas
Storage blocks	9 blocks, 384 container positions per block
RTG-type cranes	9 independently controlled handling resources
Pickup points	16 points per loading area for 20-foot containers (144 in total); 8 paired positions per loading area for 40-foot containers (72 in total)
Total storage capacity	3,456 twenty-foot containers
Measurement objects	Truck time log, counters, occupancy logs and crane-state statistics

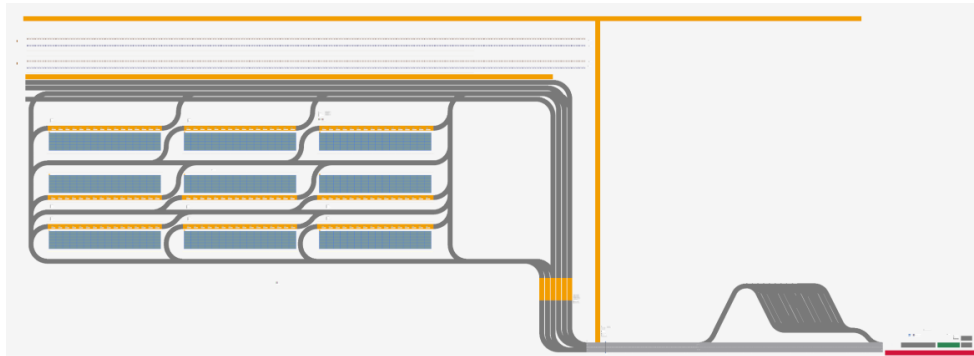


Figure 1. Overall structure of the authors' intermodal terminal simulation model developed in Tecnomatix Plant Simulation

3.2. Data structures and experiment control

The model uses table-based data management to connect containers, trucks and storage positions. The container-arrival table stores identifiers, priority and special-handling data, size, storage coordinate, arrival and departure times, weights, load status and truck assignment. Operational data are therefore separated from graphical objects and can later support rule-based or AI-assisted storage allocation.

Measurement and control objects are also separated. Sensors identify entering trucks; time logs and counters record vehicle flow; occupancy variables describe checkpoint and checkout states; and initialization clears task lists and measurement tables, resets variables and moves cranes to their home positions. The full-capacity experiment stops when all nine blocks reach 384 containers. Clear data ownership and reproducible initialization reduce hidden dependencies between experiments.

4. PROGRAMMING REFINEMENT

4.1. Generalization of repeated control logic

The first programming problem was the presence of several object groups that operate according to the same rules. The loading areas are labelled A to I, and each loading area contains 16 local pickup points for 20-foot containers, giving 144 individual 20-foot pickup points in total. For 40-foot containers, these are handled as eight paired pickup positions per loading area, giving 72 40-foot positions in total. Checkpoint and checkout resources also share similar reservation procedures. Writing an independent condition chain for every area, point or resource would produce long methods containing many nearly identical branches. Such code is difficult to review because a correction must be repeated consistently in every branch, and it also becomes a barrier to expansion when additional areas or resources are introduced.

The generalized structure therefore uses loops, generated names and calculated indices. An area index identifies one of the nine loading areas (A-I), while a local pickup-point index $p = 1, \dots, 16$ identifies one of the sixteen 20-foot pickup points within that area. For 40-foot handling, adjacent 20-foot points are treated as pairs, so paired position $s = 1, \dots, 8$ corresponds to local points $p_1 = 2s - 1$ and $p_2 = 2s$. Thus, paired position 1 uses points 1 and 2, position 2 uses points 3 and 4, and position 8 uses points 15 and 16. The relevant object names can therefore be generated from the area label and local indices instead of being selected through separate hard-coded branches. The approach can also be applied to crane identifiers and storage-block coordinates. For the pickup-point logic, a fully object-specific structure could require up to 144 separately maintained 20-foot point references, whereas the indexed formulation evaluates one common rule over the 9×16 address space; the same structure naturally represents the 9×8 paired positions used for 40-foot containers. This is a quantitative structural simplification of the addressing scheme; it is not presented as a measured line-count or runtime reduction.

4.2. Reservation states and resource consistency

The second refinement area concerns shared resources. Trucks reserve checkpoint and checkout positions before moving toward them and release the positions after leaving. Delayed reservation can allow two trucks to select the same resource, whereas a missing release can leave the resource falsely occupied. The same paired-state principle is used for crane tasks: a crane is activated before task acceptance and released after completion or cancellation. Keeping these transitions in clearly identifiable method sections reduces inconsistent states and simplifies error recovery.

4.3. Limiting blocking conditions and unnecessary synchronization

Wait conditions are useful when an object must remain stationary until a required state becomes true, but excessive blocking can create model-level deadlocks. A truck may wait for a pickup position while the crane that would empty that position waits for a condition indirectly dependent on the same truck. The refined logic therefore separates truck, crane and storage completion rules. Once a container is placed at a valid intermediate pickup point and

responsibility has passed to the crane task list, the truck can proceed toward checkout instead of waiting for final storage.

Necessary waits should remain local and have a clearly identifiable releasing event. Diagnostic counters or timestamps help distinguish genuine congestion from a blockage produced by control logic.

4.4. Initialization, logging and modular verification

Repeatable experiments require an identical initial state. A central method clears crane task lists, logs and measurements, resets counters and occupancy variables, and returns the nine cranes and their lifting devices to defined starting positions. Logging is separated from operational control, so truck times, vehicle counts, checkpoint occupancy and crane states can be extended without rewriting transport logic. Verification first used a reduced three-crane, 48-position model before the same programming patterns were transferred to the complete nine-block model. These values provide objective information about the scale over which the initialization and control principles were applied, although initialization time and peak memory were not recorded (Table II).

Table II.
Programming refinement measures and their evaluation rationale

Development measure	Primary problem addressed	Intended contribution
Loops, generated names and calculated indices	Repeated object-specific conditions	Fewer repeated rule locations; consistent modification and easier expansion
Explicit reserve-release state pairs	Resources remaining falsely occupied	Consistent checkpoint, checkout and crane-state handling
Decoupled truck and crane completion rules	Artificial circular waiting dependencies	Reduced risk of artificial deadlock and unnecessary synchronization
Central initialization method	Residual states from earlier experiments	Defined and repeatable starting conditions
Separate operational and measurement objects	Control logic mixed with data logging	Clearer debugging and easier extension of indicators
Small-model verification before scaling	Errors hidden by model size	Earlier identification of logic and data-collection problems

5. GRAPHICAL REFINEMENT

5.1. Role of three-dimensional visualization

Graphical representation supports verification, communication and presentation. Routes, delayed vehicles, crane movements, storage occupancy and traffic directions should remain visible, but fine surface geometry and small structural components are useful only when they serve a specific research, verification or communication task. This is consistent with simulation-visualization studies showing that 3D displays can improve error detection and

model understanding [15], while model-fidelity studies emphasize selecting detail according to the intended analysis and validation purpose [16]. Functional information is therefore more important than maximum geometric realism (Fig. 2).

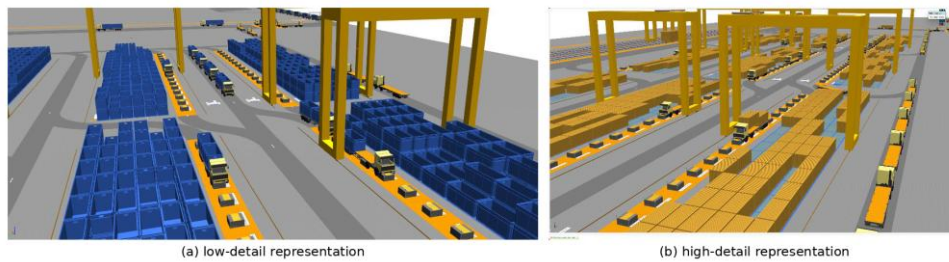


Figure 2. Comparison of low- and high-detail graphical representations created by the authors in Tecnomatix Plant Simulation

5.2. Low-detail and high-detail representations

Figure 2 compares simple and detailed container representations. Low-detail geometry keeps large stacks visually uniform and supports rapid observation during debugging and repeated experiments. High-detail geometry improves realism, but the additional complexity is repeated hundreds or thousands of times. A practical solution is to use simple objects for routine runs and activate detailed scenes only for selected demonstrations or publication images.

5.3. Practical criteria for selecting graphical detail

The level of graphical detail should be selected using observable criteria rather than realism alone. High detail is justified when geometry affects the represented process (for example working envelopes, clearances, stacking levels or route interpretation), when a visual feature is required to verify model behaviour, or when a specific view is needed for communication with stakeholders. Low detail is preferred when an object is replicated many times, moves continuously, or contains fine geometry that does not influence simulation logic or the interpretation of results. A practical decision rule is to start with the lowest detail that preserves functional interpretation and add geometry only when removing that detail would impair verification, spatial understanding or the communication objective. For highly repeated objects such as containers, the repetition count should be treated as a multiplier of graphical complexity; if visualization performance is important, alternative representations should be compared in a pilot scene using file size, initialization time, peak memory and frame rate as measurable indicators. This criterion-based approach makes graphical fidelity a configurable modelling decision rather than a fixed feature of the model.

5.4. Imported CAD and external 3D objects

External CAD and 3D objects can improve realism but may contain dense meshes, hidden components and materials irrelevant to the simulation. Geometry should be simplified before import, and repeated objects deserve particular attention. Simplification must preserve

dimensions, working envelopes, stacking levels and other information that influences process behaviour; refinement removes non-functional detail rather than changing physical assumptions.

6. EVALUATION AND DISCUSSION

6.1. Effects on maintainability and extensibility

The programming measures primarily change model structure rather than the operational logic being studied. Generic methods reduce the number of locations in which the same rule is implemented, so selection or reservation logic can be modified once and then checked across all similar objects. In the case study, one indexed addressing principle covers nine loading areas and 144 pickup points for 20-foot containers, while the same structure represents 72 paired pickup positions for 40-foot containers; paired state transitions are applied consistently to shared checkpoint, checkout and crane resources. This provides structural evidence for maintainability and extensibility at the scale of the implemented model and is consistent with literature linking separation of concerns and model continuity to modifiability and maintainability [12,13]. The reduced three-crane verification model and central initialization procedure also follow the staged checking logic recommended in simulation verification practice [14]. However, code-line counts, defect rates and versioned before-and-after implementations were not recorded, so the paper does not treat shorter development time, fewer errors or faster execution as quantitatively demonstrated outcomes.

6.2. Effects on experiment execution and visualization

Graphical simplification is expected to reduce file size, memory use and rendering demand, but the magnitude was not measured in the original experiments and will depend on hardware, software version, object geometry and the number of repeated objects. The literature supports a genuine trade-off: 3D visualization can aid verification and model understanding [15], while higher fidelity increases modelling effort and may introduce additional computational cost [16]. Graphics should therefore be treated as a configurable experimental layer: low detail for routine runs and debugging, and selective high detail when geometry is functionally or communicatively necessary. A future controlled benchmark should compare otherwise identical model versions and record file size, initialization time, wall-clock execution time, peak memory use and frame rate while holding random seeds, process logic and output data constant. Until such a benchmark is performed, computational benefits remain hypotheses rather than measured results.

6.3. Practical development guidelines

A practical sequence emerges from the case study: create a reduced test model; define naming and indexing rules before scaling; use paired state transitions for shared resources; ensure that every blocking condition has a known releasing event; implement centralized initialization and logging before comparative experiments; and select graphical fidelity using functional relevance, repetition count, motion frequency, verification need and presentation purpose. Consistent naming and centralized methods also support collaborative review and reduce dependence on the memory of the original developer. These steps should be

understood as development guidance rather than a universally optimal recipe; the appropriate implementation must remain consistent with the modelling objective and the behaviour of the represented system.

6.4. Limitations

The case study represents a generalized terminal rather than a calibrated facility. The original work did not preserve a complete sequence of pre-refinement model versions or controlled runtime, memory, file-size and initialization measurements. Consequently, the paper evaluates structural and methodological effects and does not attribute operational scenario improvements directly to the programming or graphical measures. Transferability also has limits. The programming principles are most directly applicable to object-based discrete-event models with repeated resources, explicit states and rule-based control; the exact naming, indexing and method syntax are specific to Tecnomatix Plant Simulation and must be adapted in other simulation environments. Reserve-release and blocking-condition guidance is relevant where resources are represented by discrete occupancy states, but it may require reformulation for agent-based, continuous, physics-based or strongly human-centred simulations. The graphical principles are more broadly transferable, yet high geometric fidelity may be essential when clearances, collisions, ergonomics, safety zones or immersive visualization are part of the research question. The framework should therefore be viewed as a set of transferable decision principles whose effects must be revalidated for each model class and industrial application. Future work should benchmark versioned models under identical hardware conditions and vary object population, 3D complexity and logging frequency to establish quantitative thresholds for when each measure becomes beneficial.

7. SUMMARY

This paper examined programming and graphical refinement principles in Tecnomatix Plant Simulation through an intermodal container-terminal case study. The general methodological lessons are to replace repeated object-specific control with indexed rules, manage shared resources through explicit paired state transitions, keep synchronization local to the physical dependency being represented, initialize experiments from a defined common state, separate measurement from operational control, and treat 3D fidelity as a configurable layer selected according to functional and communication needs. These principles are intended to improve transparency, maintainability, extensibility and repeatability, but their computational effects should be verified with controlled measurements rather than assumed. The case-specific evidence is the successful application of the framework to nine loading areas, 144 pickup points for 20-foot containers (corresponding to 72 paired positions for 40-foot containers), nine storage blocks, nine RTG-type cranes and 3,456 twenty-foot container positions, preceded by verification on a reduced three-crane, 48-position model. For the pickup-point logic, one indexed rule addresses the 9 x 16 20-foot location space and its 9 x 8 paired 40-foot representation instead of maintaining point-specific selection branches. Because no versioned hardware-independent benchmark was recorded, the study does not claim a numerical speed-up, memory reduction or shorter initialization time. The principal contribution is therefore a practice-oriented methodological framework and a set of objective criteria that can be tested quantitatively in future Plant Simulation and other object-based discrete-event models.

ACKNOWLEDGEMENTS

Project No. 2023-1.2.4-TÉT-2023-00027 has been implemented with the support provided by the Ministry of Culture and Innovation of Hungary from the National Research, Development and Innovation Fund, and was financed under the 2023-1.2.4-TÉT funding scheme.

REFERENCES

- [1] Carteni, A. & De Luca, S. (2012). Tactical and strategic planning for a container terminal: Modelling issues within a discrete event simulation approach. *Simulation Modelling Practice and Theory* **21**(1), 123-145, <https://doi.org/10.1016/j.simpat.2011.10.005>
- [2] Capodilupo, L., Ricci, S. & Tombesi, E. (2016). Discrete events simulation of intermodal terminals operation: Modelling techniques and achievable results. *Civil-Comp Proceedings* **110**, 1-13, <https://doi.org/10.4203/ccp.110.288>
- [3] Srisurin, P., Pimpanit, P. & Jarumaneeroj, P. (2022). Evaluating the long-term operational performance of a large-scale inland terminal: A discrete event simulation-based modeling approach. *PLOS ONE* **17**(12), e0278649, <https://doi.org/10.1371/journal.pone.0278649>
- [4] Neagoe, M., Hvolby, H. H., Taskhiri, M. S. & Turner, P. (2021). Using discrete-event simulation to compare congestion management initiatives at a port terminal. *Simulation Modelling Practice and Theory* **112**, 102362, <https://doi.org/10.1016/j.simpat.2021.102362>
- [5] Muravev, D., Hu, H., Rakhmangulov, A. & Mishkurov, P. (2021). Multi-agent optimization of the intermodal terminal main parameters by using AnyLogic simulation platform: Case study on the Ningbo-Zhoushan Port. *International Journal of Information Management* **57**, 102133, <https://doi.org/10.1016/j.ijinfomgt.2020.102133>
- [6] Gambardella, L. M., Rizzoli, A. E. & Zaffalon, M. (1998). Simulation and planning of an intermodal container terminal. *Simulation* **71**(2), 107-116, <https://doi.org/10.1177/003754979807100205>
- [7] Swisher, J. R., Jacobson, S. H. & Yücesan, E. (2003). Discrete-event simulation optimization using ranking, selection, and multiple comparison procedures: A survey. *ACM Transactions on Modeling and Computer Simulation* **13**(2), 134-154, <https://doi.org/10.1145/858481.858484>
- [8] Swisher, J. R., Hyden, P. D., Jacobson, S. H. & Schruben, L. W. (2004). A survey of recent advances in discrete input parameter discrete-event simulation optimization. *IIE Transactions* **36**(6), 591-600, <https://doi.org/10.1080/07408170490438726>
- [9] Jachimowski, R. & Kłodawski, M. (2025). The impact of container yard layout on the cargo handling time of external transport vehicles in an intermodal terminal. *Archives of Transport* **73**(1), 179-193, <https://doi.org/10.61089/aot2025.4f22k916>
- [10] Cavada, J. P., Cortés, C. E. & Rey, P. A. (2023). Comparing allocation and relocation policies at a logistics service container terminal: A discrete-event simulation approach. *Central European Journal of Operations Research* **31**(4), 1281-1316, <https://doi.org/10.1007/s10100-023-00857-1>
- [11] Ma, N., Zhou, C. & Stephen, A. (2021). Simulation model and performance evaluation of battery-powered AGV systems in automated container terminals. *Simulation Modelling Practice and Theory* **106**, 102146, <https://doi.org/10.1016/j.simpat.2020.102146>
- [12] Cetinkaya, D., Verbraeck, A. & Seck, M. D. (2015). Model continuity in discrete event simulation: A framework for model-driven development of simulation models. *ACM Transactions on Modeling and Computer Simulation* **25**(3), Article 17, 1-24, <https://doi.org/10.1145/2699714>
- [13] Blas, M. J. & Gonnet, S. (2021). Computer-aided design for building multipurpose routing processes in discrete event simulation models. *Engineering Science and Technology, an International Journal* **24**(1), 22-34, <https://doi.org/10.1016/j.jestech.2020.12.006>
- [14] Sargent, R. G. (2010). Verification and validation of simulation models. *Proceedings of the 2010 Winter Simulation Conference*, 166-183, <https://doi.org/10.1109/WSC.2010.5679166>

- [15] Akpan, I. J. & Brooks, R. J. (2014). Experimental evaluation of user performance on two-dimensional and three-dimensional perspective displays in discrete-event simulation. *Decision Support Systems* **64**, 14-30, <https://doi.org/10.1016/j.dss.2014.04.002>
- [16] Nigischer, C., Reiterer, F., Bougain, S. & Grafinger, M. (2023). Finding the proper level of detail to achieve sufficient model fidelity using FlexSim: An industrial use case. *Procedia CIRP* **119**, 1240-1245, <https://doi.org/10.1016/j.procir.2023.02.192>
- [17] Siemens Digital Industries Software: Tecnomatix Plant Simulation software. Retrieved from: <https://plm.sw.siemens.com/en-US/tecnomatix/plant-simulation-software/> (accessed: 6 May 2026).