Proceedings of the 2025 Winter Simulation Conference
E. Azar, A. Djanatliev, A. Harper, C. Kogler, V. Ramamohan, A. Anagnostou, and S. J. E. Taylor, eds.

A FEDERATED SIMULATION FRAMEWORK FOR MODELING COMPLEX LOGISTICS: AMAZON SORT CENTER CASE STUDY

Rahul Ramachandran[1], Hao Zhou[1], and Anish Goyal[1]

[1]Worldwide Design Engineering, Amazon, Bellevue, WA, USA

ABSTRACT

Amazon's sort centers represent highly complex systems where traditional monolithic simulation approaches fall short due to computational limitations, extended time-to-value, and challenging model maintenance. This case study presents a federated simulation framework that decomposes the sort center model into four key federates: inbound operations, main sortation and outbound operations, non-conveyable processing, and non-compliant item handling. Each federate operates independently while maintaining synchronization through a conservative time-stepping algorithm and structured message-passing protocol. Our implementation demonstrates significant advantages, including a 40% reduction in model development time, 15% improvement in model execution time, and improved scenario testing capabilities. This approach enables early optimization of sort center design, identification of cross-process bottlenecks, and better first-time launch quality. This real-world application showcases the practicality and benefits of federated simulation in modern logistics, offering valuable insights for practitioners modeling complex industrial systems.

1 BACKGROUND

Amazon's sort centers process thousands of packages daily through multiple process paths such as regular sortation, non-conveyable items, and non-compliant packages. These facilities incorporate a complex mix of automated systems (including Autonomous Mobile Robots, conveyors, sorters, and sophisticated warehouse management and control software) alongside manual processes. Simulating these complex operations traditionally required monolithic models that were difficult to develop, validate, and maintain. Key challenges included long development cycles requiring cross-team coordination, complex model validation across multiple processes, difficulty in parallel development and testing, and resource-intensive computation for full-scale simulation. Additionally, monolithic simulations required developers to be subject matter experts across all systems, creating knowledge bottlenecks and ownership challenges. These limitations led to longer project timelines and compromised decision-making capabilities.

2 SOLUTION

Our federated simulation framework decomposes complex systems into separate discrete event simulations (federates) that operate independently while maintaining synchronized execution. This distributed approach is orchestrated through a federation server including three essential services: the Message Broker handles message delivery between federates; the Clock Service receives federate's simulation clock update and publishes time horizon to federates; the State Manager receives federate's state update and publishes state transition to federates. Each federate includes two essential components: Federation Client and FlexSim Model. The Federation Client handles communications with the three services within the Federation Server and controls the simulation engine to execute time horizon and station transition. It also provides APIs to the FlexSim Model for message exchange with other federates. Messages are communicated through a JSON-based universal communication library that supports various data types, enabling standardized information exchange between federates. The framework utilizes standardized interfaces for communication and deployment, supporting parallel development and testing of individual federates while maintaining overall simulation integrity.

3 USE CASES AND BENEFITS

We decomposed the complex Amazon sort center operations simulation into four distinct federates:

1. Inbound Operations federate simulates receiving and unloading processes, including trailer docking, package unloading, and initial sortation decisions.
2. Main Sortation and Outbound Operations federate represents the primary package flow through automated conveyor systems, sorters, and outbound trailer loading
3. Non-conveyable Processing federate models the handling of over sized and irregular items through specialized equipment and manual processes
4. Non-compliant Handling federate simulates the processing of packages requiring special handling or problem-solving

The federated framework provides valuable benefits both pre- and post-launch, enabling validation of processes and optimization of resources during initial planning while supporting continuous improvement through bottleneck analysis and performance prediction after implementation. A key advantage of this architecture is its support for parallel development and the ability to model specific operations in great detail. For instance, the Inbound Operations federate could be refined to consider intricate details such as associate walking paths, individual package handling based on size and shape, and precise unloading times. This level of detail, when combined with the broader system view, provides a more accurate representation of the entire operation without compromising on development speed or overall model performance. Quantifiable benefits include 40% reduction in model development time, 15% improvement in simulation run time, improved accuracy in throughput predictions, enhanced ability to test operational scenarios, and better first-time launch quality metrics.

4 CHALLENGES

Key challenges addressed during implementation included maintaining temporal causality across federates, ensuring efficient state synchronization, managing different time scales across processes, and ensuring model consistency. While the framework successfully reduced development time through parallel development, the runtime performance can be a consideration, particularly for smaller size models. For these simpler scenarios, the framework may introduce some overhead compared to simplified monolithic models due to the necessary external services synchronizing simulation clocks and managing information exchange between federates. However, this overhead becomes less significant as model complexity increases, and is outweighed by the benefits in development time and flexibility for larger, more complex simulations. A significant challenge emerged in debugging scenarios, particularly when tracing issues across connected processes spanning multiple federates. Implementation challenges included defining appropriate federate boundaries, establishing standard interfaces, validating individual and integrated performance, and managing data dependencies.