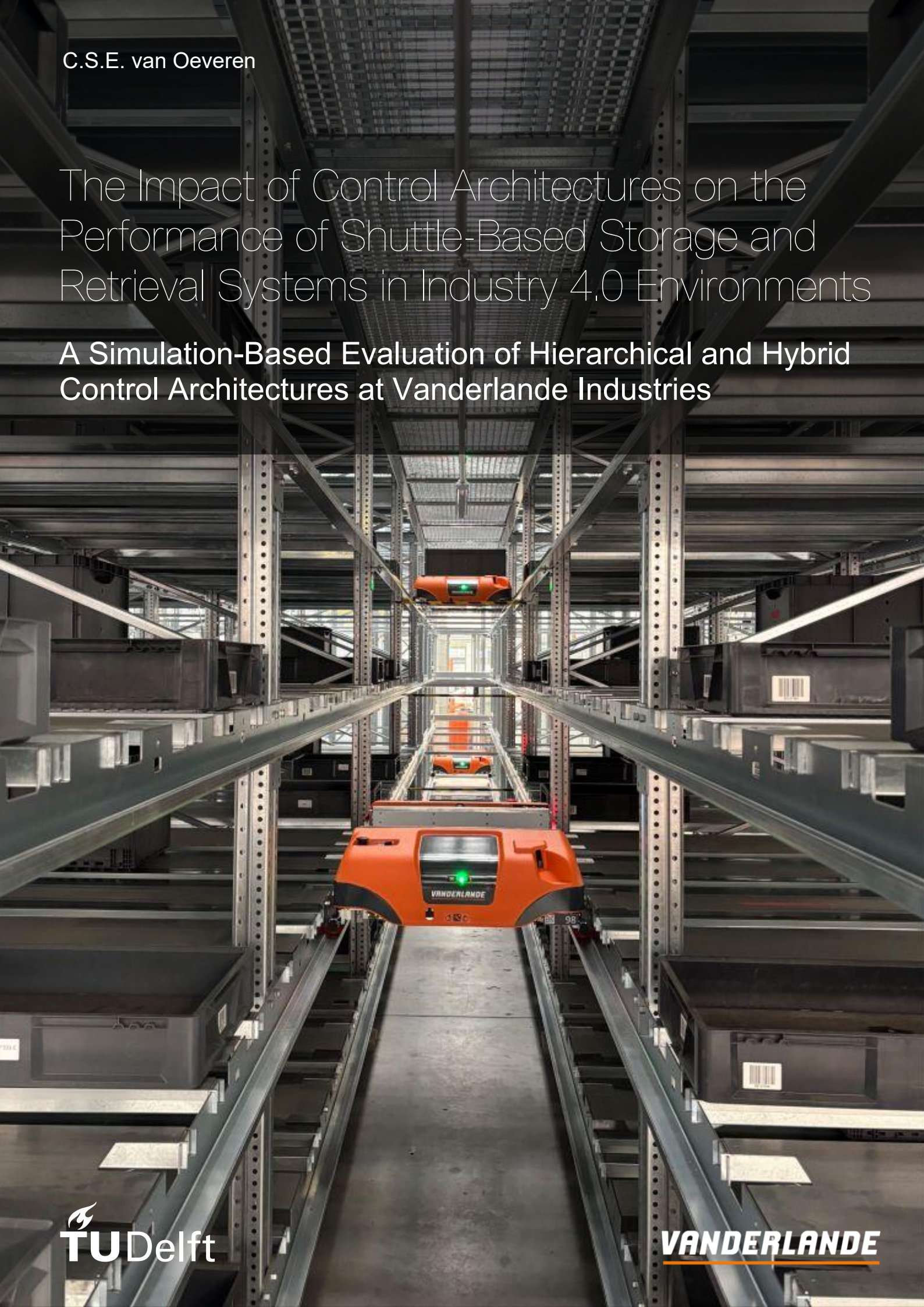
C.S.E. van Oeveren

The Impact of Control Architectures on the Performance of Shuttle-Based Storage and Retrieval Systems in Industry 4.0 Environments

A Simulation-Based Evaluation of Hierarchical and Hybrid Control Architectures at Vanderlande Industries



TU Delft

VANDERLANDE

The Impact of Control Architectures on the Performance of Shuttle-Based Storage and Retrieval Systems in Industry 4.0 Environments

A Simulation-Based Evaluation of Hierarchical and Hybrid Control Architectures at Vanderlande Industries

By

C.S.E. van Oeveren

Master Thesis

in partial fulfilment of the requirements for the degree of

Master of Science
in Mechanical Engineering

at the Department Maritime and Transport Technology of Faculty Mechanical, Maritime and Materials Engineering of Delft University of Technology
to be defended publicly on Friday September 5, 2025 at 10:00 AM

Cover Photo [40]

Student number:	4591046	
MSc track:	Multi-Machine Engineering	
Report number:	2025.MME.9086	
Thesis committee:	Prof.dr. R.R. Negenborn,	TU Delft, chair
	Dr.ir. Y. Pang,	TU Delft, supervisor
	H. Huijsman,	Vanderlande Industries, supervisor
	J. García Martín,	TU Delft, committee member
Date:	Month 09, 2025	

An electronic version of this thesis is available at http://repository.tudelft.nl/.

It may only be reproduced literally and as a whole. For commercial purposes only with written authorization of Delft University of Technology. Requests for consult are only taken into consideration under the condition that the applicant denies all legal rights on liabilities concerning the contents of the advice.

TU Delft

Preface

Dear reader,

After years of lectures, projects, deadlines and more caffeine and late nights than I'd like to admit, I've reached the final chapter of my time as a student. The thesis in front of you is the last piece of my master's journey in Multi-Machine Engineering at TU Delft and it's been quite a ride.

I began this project ready for both the technical challenges and the steady perseverance needed to complete it. There were weeks spent immersed in research papers, months spent attempting to get a stubborn simulation to run at last and periods where progress felt slow. An unexpected test of perseverance came when I had to take a break to focus on my health. However, it never felt as a matter of if I would finish but of when I could give it my full attention again. Getting back into it was daunting but also genuinely exciting. I'm grateful to everyone who supported me, encouraged me and gave me the time and space to ease back into the work. And now here it is, a finished work of which I am truly proud.

One of the highlights of the whole process was seeing the first full set of simulation results come in. After countless weeks of designing, coding and tweaking, the simulation was finally ready. It was very exciting to see that the first results confirmed my initial expectations and hopes that the new architectural design that I had been working on would create a positive impact.

I could not have done this without the support of everyone around me. Thank you to my supervisors, Yusong, Hessel and Niels for your guidance, feedback, and positivity throughout the project. Thank you to Vanderlande Industries and the DfX team for providing the case study context which made this research more concrete and relevant to real-world applications. Thank you to my friends and family for listening to my stories about shuttles, control architectures and simulations. A special thanks to my parents and housemates who saw both the good and the bad during this process and never stopped believing in me.

Finishing this thesis feels like closing a long but incredibly rewarding chapter. It's been full of challenges but also full of learning and victories along the way. I hope this thesis reflects that.

Enjoy the read,

C.S.E. van Oeveren
05 September 2025

Executive Summary

Modern warehouse systems are undergoing a fundamental transformation. This change is driven by the growing demands for rapid, large-volume order fulfillment and by the evolution of Industry 4.0 (I4.0) capabilities. Shuttle-Based Storage and Retrieval Systems (SBSRS) are critical systems in a warehouse thanks to their ability to perform parallel storage and retrieval operations via fleets of shuttles operating on fixed rails. SBSRS are well-suited for dynamic, high-density environments such as e-commerce fulfillment centers [45]. Literature research suggests that SBSRS conventionally operate under centralized or hierarchical control architectures. These control designs manage task allocation and path planning through a central controller. These conventional architectures prove effective in controlled settings yet they create major challenges when operating in dynamic and large-scale environments. In such environments they cause congestion at shared resources, coordination bottlenecks and limited adaptability [53, 74, 62].

The evolution of warehousing systems under I4.0 requires supporting technologies like Cyber-Physical Systems (CPS), the Internet of Things (IoT) and Artificial Intelligence (AI). This leads to design principles that focus on decentralization and autonomy and real-time responsiveness [85, 37]. The I4.0 capabilities do not match the rigid top-down control logic of current SBSRS, which call for a rethink of the architectural foundation of SBSRS control systems. Current research offers limited insight into how I4.0-inspired control architectures perform in SBSRS.

This study evaluates the impact of alternative control architectures on the operational performance of SBSRS with a particular focus on I4.0-oriented designs. The research investigates whether a hybrid control architecture using intelligent shuttle agents capable of local decision-making and peer-to-peer coordination, can outperform conventional top-down architecture. The two control architectures are compared by assessing their impact on the operational performance of the system. This is done by assessing throughput, operational efficiency, scalability and robustness under varying system scales and shuttle densities. The research is guided by the following question:

What is the impact of control architectures on the operational performance of a shuttle-based storage and retrieval system?

To answer this question, this study designs, implements and evaluates two control architectures for SBSRS. The two conceptual control architectures were developed using Multi-Agent System (MAS) technology: (1) a hierarchical system in which a central controller assigns tasks and pre-computes routes and (2) an I4.0-inspired hybrid system where shuttles act autonomously using local congestion-awareness, decentralized routing and deadlock recovery [86, 41]. The conceptual model of the hybrid architecture and therefore its simulation, is designed in line with five improvement needs identified through literature and system-level analysis: (1) distributed and autonomous decision-making, (2) adaptive traffic and resource management, (3) enhanced inter-agent coordination, (4) modular and scalable architecture and (5) I4.0 readiness. These improvement needs provided essential design targets for rethinking the SBSRS control architecture.

To compare and evaluate, both architectural designs are implemented in a simplified yet representative use case based on the Vanderlande ADAPTO system. The implemented model includes a simplified but representative baseline configuration with standardized components, ensuring a controlled environment for scientifically meaningful comparisons. A simulation was developed to evaluate and compare the impact of the control architectures on the operational system performance. This was established using a hybrid Discrete-Event Simulation (DES) and Agent-Based Modeling (ABM) framework. The operational system performance is assessed using four core metrics: throughput (capacity), normalized efficiency (normalized throughput per distance unit), scalability (sensitivity to size and density) and robustness (performance variance). To evaluate the impact of control architecture under different operating conditions, experimental scenarios varied along two primary dimensions: system size

(10×10, 15×15, 20×20 and 25×25 - number of aisles x number of storage slots)) and shuttle density (1 to 6 concurrent agents). The former measures routing complexity and architectural scalability and the latter tests the system's ability to manage congestion and coordinate access to shared resources.

The simulation results demonstrate that control architecture selection strongly affects SBSRS performance. The hybrid model consistently outperforms the hierarchical architecture across all performance metrics and scenarios. The performance gap was most pronounced in large-scale environments operating with a high number of concurrent shuttles. A quantitative analysis and comparative evaluation of the hybrid control architecture against the hierarchical lead to the following quantitative results:

- Throughput: Up to 90% higher in large-scale, high-density systems; 30-60% higher in mid-sized configurations; up to 36% higher under optimal comparable conditions.
- Efficiency: Up to 68% improvement in normalized throughput (tasks per distance unit).
- Scalability: Sustained or improving performance as system size and shuttle count increased, unlike hierarchical control which often degraded beyond two or three shuttles.
- Robustness: Significantly lower variance across replications, with hybrid control eliminating the low-outlier runs seen under centralized coordination.

This research shows that for SBSRS, adopting a hybrid control architecture can significantly improve operational performance compared to conventional hierarchical control. The hybrid approach, which combines decision-making at the shuttle level with coordinated agent interaction, achieved higher throughput, efficiency, sustained scalability and greater robustness across a range of operating conditions. The performance advantage was most pronounced in large-scale, high-density scenarios, highlighting the architecture's suitability for complex, high-demand environments and its potential as a feasible path toward next-generation, I4.0-ready warehouse control systems. The research findings fill an essential knowledge gap regarding architectural comparisons and Industry 4.0 principle implementation in the field of SBSRS. They also provide warehouse designers and automation engineers with practical knowledge to enhance system throughput, efficiency, robustness and scalability. For Vanderlande Industries, the results provide simulation-based recommendations on how a hybrid control architecture can enhance the performance and future readiness of their SBSRS solutions.

The scope of the study is limited to a single level of a tier-captive SBSRS, as shuttles remain on one tier and the research scope is limited to the shuttle agent. Other system components such as vertical lifts, buffer zones and multi-level routing are excluded. The evaluation assumes ideal operating conditions with no failures, delays, or disturbances. The control logic for both architectures was intentionally simplified to ensure a fair and transparent comparison. Lastly, no empirical validation was conducted due to the lack of suitable benchmark data for the scope of this study.

For future work it is recommended to extend the current model to multi-level SBSRS configurations with vertical lifts, buffering zones and complex routing layers, which are expected to further enhance the benefits of hybrid control. Next to that, stochastic disturbances should be added to the model like equipment failures or communication noise to enable more realistic robustness evaluation, where hybrid architectures are likely to excel. Additional research on agent coordination, learning mechanisms and communication strategies is suggested to further improve MAS performance. Furthermore, empirical validation through digital twins or real-world pilots is recommended to assess implementation feasibility. Lastly, a cost-benefit analysis is recommended to cover capital expenditures, operational expenses and return on investment which is necessary to evaluate the practical viability of hybrid control in industry.

List of Abbreviations

ABM	Agent-Based Modeling
AMR	Autonomous Mobile Robot
AI	Artificial Intelligence
ASRS	Automated Storage and Retrieval System
CA-Safe	Cross-Aisle Safe zone
CI	Confidence Interval
COL	Closest Open Location
CPS	Cyber-Physical System
DES	Discrete-Event Simulation
DO	Design Objective
ERP	Enterprise Resource Planning
FIFO	First-In-First-Out
FCFS	First-Come-First-Served
GA	Genetic Algorithm
I4.0	Industry 4.0
IN	Improvement Need
IoT	Internet of Things
KPI	Key Performance Indicator
MAS	Multi-Agent System
MCDM	Multi-Criteria Decision Making
P&D	Pick-up and Deposit
RFID	Radio-Frequency Identification
SBSRS	Shuttle-Based Storage and Retrieval System
SPT	Shortest Processing Time First
STD	Shortest Travel Distance
TSU	Transportable Storage Unit
WCS	Warehouse Control System
WMS	Warehouse Management System

Contents

1 Introduction

The rapid development of supply chain and logistics systems has brought significant changes to warehouse operations over the past few decades [61]. Warehouses have become central to modern logistics due to the rise of e-commerce and the increased demand for fast and efficient order fulfillment. To keep up, warehouse processes now demand continuous automation and optimization to accommodate high-volume, dynamic workflows. The COVID-19 pandemic further accelerated this transformation, increasing reliance on storage services and digital logistics solutions [35]. Together, these developments highlight the need for more flexible, intelligent and scalable warehouse systems. In response, modern warehouses are transitioning to smart warehousing, integrating automation, control, and communication technologies to enhance efficiency and scalability [91].

A central technology in smart warehouses is the Automated Storage and Retrieval Systems (ASRS), which automates the storage and retrieval of goods through robotic shuttles, conveyors, lifts and control systems. These systems minimize human intervention while improving inventory accuracy, throughput, and space utilization [33]. The effectiveness of ASRS depends not only on their mechanical design but also on the control mechanisms and control architecture that govern their operations. ASRS function in highly mechanized and computer-controlled environments, where system components like transportable storage units, retrieval mechanisms, and intelligent routing systems work together [67]. According to De Koster et al., the efficiency of ASRS relies on how well key functions such as task allocation, congestion management, and decision-making, are coordinated [12].

Within the broader ASRS category, this research focuses on Shuttle-Based Storage and Retrieval Systems (SBSRS), a specific ASRS configuration where automated shuttles operate on rails throughout the system to perform storage and retrieval operations. SBSRS are widely used in high-throughput, space-constrained environments such as e-commerce fulfillment centers and automated warehouses [53]. These systems rely on multiple concurrently operating shuttles, making the system's control architecture a critical component of overall performance, particularly in coordinating resources and in handling congestion.

Through literature research, it is established that control systems in SBSRS conventionally rely on centralized or hierarchical architectures. In such architectural design, a central or higher-level control unit dictates task assignments, path planning, and congestion resolution. While centralized control can be effective in structured, low-complexity systems, it presents limitations in SBSRS due to the system's dynamic nature, parallelism, and the need for responsive decision-making across multiple shuttles. Centralized and hierarchical architectures presents challenges in scalability, adaptability, and fault tolerance [28, 27]. Moreover, as warehouse operations expand, these control architectures struggle with increased computational loads, reduced responsiveness to unexpected disruptions, and a higher risk of single-point failures [39]. These limitations question whether centralized and hierarchical architectures can meet the needs of modern dynamic warehouse environments.

The rise of Industry 4.0 (I4.0) is driving the evolution of warehousing systems, making the architectural limitations of current designs increasingly apparent. I4.0 refers to the fourth industrial revolution, characterized by the convergence of physical and digital systems. I4.0 technologies such as the Internet of Things (IoT), Artificial Intelligence (AI), and Cyber-Physical Systems (CPS) introduce key principles like distributed control, real-time communication, and local autonomy in physical and digital systems. These capabilities conflict with the rigid, top-down nature of conventional SBSRS control. This misalignment creates an urgent need to re-evaluate control architectures to ensure compatibility with the flexibility, modularity, and intelligence required in modern warehouse environments [85]. Integrating I4.0 capabilities into SBSRS could unlock greater responsiveness and adaptability and can transform the storage and retrieval operations from static, top-down systems into adaptive, intelligent environments [85].

While the integration of I4.0 holds significant potential for SBSRS, current academic and industrial designs predominantly rely on centralized or hierarchical control architectures, as shown in recent literature. The architectural implementation of integrating these technologies into SBSRS and the implementation of I4.0 principles like distributed control have not been extensively studied in the field of SBSRS [89, 14]. Existing academic work has extensively explored improvements in individual control logics like scheduling, routing and storage policies, all within conventional control frameworks. The underlying control architecture itself has rarely been challenged. As a result, there is limited understanding of how architectural redesign, especially toward decentralization and agent-based paradigms, can affect SBSRS performance.

This research addresses this gap by evaluating how different control architectures, specifically hierarchical and hybrid multi-agent architectures, impact SBSRS performance. The evaluation is set in environments enabled by I4.0 technologies, such as real-time sensing, distributed intelligence, and cyber-physical integration. The study explores how decentralization and agent-based control mechanisms can improve throughput, operational efficiency, scalability and robustness. This research aims to serve as a foundational step toward intelligent, adaptive SBSRS while leveraging the full potential of I4.0 technologies.

1.1. Problem Definition

The growing adoption of I4.0 in manufacturing and logistics is transforming modern warehousing. Technologies like CPS, IoT, and AI have the potential to improve adaptability, efficiency, real-time responsiveness and decision-making. However, SBSRS still face fundamental limitations in control and coordination, restricting their ability to fully leverage I4.0 capabilities [59]. The rigid control structures that dominate current SBSRS prevent these technologies from being fully leveraged [59]. Before integrating I4.0 technologies, it is crucial to study the underlying control architecture that manages the system operations.

A key constraint lies in the rigid control structures that continue to dominate most SBSRS installations. These systems typically operate under centralized or hierarchical control architectures, where a single controller or higher-level decision logic dictates all shuttle movements, task assignments, and routing. This architectural design introduces several operational constraints. For instance, centralized control often imposes strict safety constraints or protocols that restrict the number of concurrently active shuttles. This leads in turn to under-utilization of system capacity and reduced throughput [53]. Additionally, task assignment and routing logic are typically rule-based. It forces shuttles to execute predefined movement patterns and affords little to no autonomy to respond to congestion or fluctuating workloads [8]. As a result, these systems struggle to scale effectively and adapt to dynamic operating conditions. This problem becomes more pronounced as warehouse systems grow in size and complexity, further exposing the scalability limitations of centralized coordination.

The ADAPTO system, developed by Vanderlande Industries, illustrates these architectural challenges in a real-world industrial context. It is a multi-shuttle, level-captive SBSRS designed to maximize throughput and storage efficiency in warehouse environments. Although technically sophisticated, the system still relies on hierarchical control, with top-down logic governing shuttle coordination. This ar-

chitecture occasionally causes bottlenecks in task coordination and limits the number of simultaneously active shuttles due to global path reservations and strict safety constraints. Also, ADAPTO's current control system shows limited responsiveness to real-time disturbances, suboptimal shuttle utilization, and constrained scalability, particularly as complexity and system size increase. These practical challenges make ADAPTO a suitable and relevant use case for investigating alternative control architectures.

Literature on smart warehousing and SBSRS systems extensively explores performance optimization, through scheduling policies, routing algorithms, and resource allocation. However, the underlying control architecture itself has received little attention. Most academic work continues to assume a centralized or hierarchical model, optimizing within that framework rather than questioning its suitability for modern warehouse demands. To guide this transition, five key improvement needs have been identified through literature and system-level analysis, in chapter 2: (1) distributed and autonomous decision-making, (2) adaptive traffic and resource management, (3) enhanced inter-agent coordination, (4) modular and scalable architecture and (5) I4.0 readiness. These needs are difficult to meet within traditional architectures and provide essential design targets for rethinking the architectural design of SBSRS control.

Performance issues such as congestion, long retrieval times, and underutilized transport capacity are often treated as isolated operational inefficiencies in SBSRS literature. However, this study adopts the view that these are symptoms of a deeper systemic limitation: the rigidity of current control architectures. Without autonomous decision-making, local responsiveness, and scalable coordination mechanisms, SBSRS cannot fully exploit the distributed intelligence made possible by I4.0 technologies.

The transition toward I4.0-enabled SBSRS requires a shift from rigid, rule-based centralized or hierarchical architectures to more autonomous, distributed coordination mechanisms. Distributed control architectures allow shuttles to make local, real-time decisions based on system conditions. This change could potentially significantly improve adaptability, resource utilization and ultimately the overall operational system performance. However, the design and implementation of such control architectures in SBSRS remain largely unexplored, both in academia and industry.

To address this gap, this study investigates how different control architectures impact SBSRS performance within an I4.0 context. By evaluating centralized, hierarchical, distributed and hybrid control approaches, the research aims to determine which control architecture best supports scalable, adaptive, and intelligent decision-making.

1.2. Research Objective and Scope

The objective of this research is to evaluate how different control architectures impact the performance of SBSRS in the context of I4.0. The study compares and evaluates two control architectures: (1) a conventional hierarchical control architecture where all shuttle decisions (task assignment, routing, coordination) are made by a higher-level controller and (2) a novel hybrid control architecture in which autonomous shuttle agents make local decisions based on real-time conditions, supported by minimal global coordination. To guide this research, the study has three main objectives: (i) to design, test and evaluate an I4.0-inspired control architecture for future SBSRS; (ii) to compare how different control architectures impact SBSRS performance; and (iii) to use Vanderlande's ADAPTO system as a real-world reference case to guide model assumptions and contextual relevance.

To meet these objectives, the study isolates control architecture as the primary variable, allowing a clear comparison between hierarchical and hybrid control designs. The operational system performance is assessed using Discrete-Event Simulation (DES) combined with Agent-Based Modeling (ABM). The performance is measured using four Key Performance Indicators (KPIs): throughput (as the primary measure of capacity), efficiency (throughput per unit of shuttle travel), scalability (sensitivity to system size and shuttle density) and robustness (performance consistency across scenarios). Due to the conceptual and architectural focus of this study, financial metrics, energy usage and human

interaction are excluded.

To support this evaluation, a reference case is derived from Vanderlande's ADAPTO system, a real-world tier-captive SBSRS widely deployed in industry. The ADAPTO system serves as a practical and representative baseline for the simulation design. While technically advanced, the system exhibits architectural limitations typical of hierarchical control: task bottlenecks, limited real-time responsiveness, and constrained scalability due to global path reservations and central coordination. These challenges make ADAPTO a suitable case for testing whether a shift in control architecture can yield performance gains.

This study focuses on the shuttle agent whereby other system components, vertical lifts, buffers, or conveyor interfaces are excluded from the scope. The decision to focus solely on the shuttle agent level is based on four considerations. First, the shuttle coordination is central to SBSRS performance and directly influences how efficiently tasks are completed. System bottlenecks such as congestion, waiting times and underutilized capacity often stem from limitations in shuttle-level control. Second, the shuttle represents the system's primary point of real-time, distributed decision-making, where the architectural contrast between centralized and hybrid control is most noticeable. Third, architectural experimentation with shuttle agents can be practically tested and implemented in real-world systems without requiring a complete overhaul of warehouse-wide control structures. Fourth, vertical lift coordination, while important, introduces additional inter-level dependencies that would confound the architectural isolation required in this study.

The system under study is a tier-captive SBSRS, where shuttles operate only within one fixed level of the system. As a result, this study concentrates on a single-level layout where all transport and coordination occurs horizontally. Focusing on the shuttle agent allows for a constrained architectural change that can be practically tested in existing systems without requiring full control system redesign.

This study generates insights that are broadly applicable, despite its scope restrictions. By focusing on the shuttle agent, the most critical subsystem for SBSRS throughput, the research provides a generalizable foundation for architectural redesign in future smart warehousing systems. It offers a controlled yet realistic framework to explore how I4.0-inspired architectures can support scalable, adaptive and high-performance systems.

1.3. Research Question

In order to address the research objectives, the main research question is as follows:

What is the impact of control architectures on the operational performance of a shuttle-based storage and retrieval system?

To answer the main research question, a set of research sub-questions has been created.

Sub-RQ 1: *What are the improvement needs that future SBSRS must satisfy?*

Sub-RQ 2: *Which control architecture is feasible to meet the system requirements for future SBSRS?*

Sub-RQ 3: *How can a hierarchical and a hybrid control architecture be designed conceptually for an SBSRS?*

Sub-RQ 4: *Which modeling approach is best suited for accurately implementing the hierarchical and hybrid control architecture?*

Sub-RQ 5: *How can the hierarchical and hybrid control architecture for SBSRS be implemented and verified in a simulation model?*

Sub-RQ 6: *What is the impact of the hybrid control architecture on the operational performance of an SBSRS compared to the hierarchical control architecture?*

1.4. Research Methodology

This research follows a structured methodology consisting of five main phases, each designed to address a specific part of the research objective and ensure scientific rigor in the design, implementation, and evaluation of the proposed control architectures:

System Analysis
The study begins with a functional analysis of SBSRS based on literature. This phase identifies key operational challenges and performance bottlenecks in current systems, particularly those arising from architectural constraints. The analysis establishes the system limitations and improvement needs that future SBSRS control architectures must address. These needs form the foundation for selecting and designing appropriate control architectures.

Literature Review
In response to the architectural limitations identified through the system analysis, this study conducts a targeted literature review to examine which control architectures are best suited to address the five previously defined improvement needs (IN1-IN5). The review focuses on centralized, hierarchical, distributed, and hybrid control structures. To facilitate this architectural comparison, the study adopts Multi-Agent Systems (MAS) as a modeling and analytical framework. A theoretical foundation is first established by introducing key MAS concepts and definitions. This is followed by a structured review that clarifies how different MAS control paradigms translate into system behavior within SBSRS environments. Each architecture type is analyzed in detail, highlighting its operational characteristics, advantages, limitations and relevance within SBSRS literature.

Conceptual Modeling
Based on insights from the system analysis and literature review, two control architecture concepts are defined using an MAS modeling approach: a conventional centralized architecture and an I4.0-inspired hybrid architecture. These conceptual models define the roles, decision logic, coordination mechanisms, and agent interactions that guide task assignment and shuttle routing. MAS is used as a conceptual design framework that supports the implementation of both architectures using consistent modeling primitives.

Simulation
The control architectures are implemented in a simulation environment using a hybrid modeling approach that combines DES for process flow and Agent-Based Modeling (ABM) for autonomous shuttle behavior. The simulation model is based on the structural layout and operating principles of a use case from Vanderlande Industries, the ADAPTO system. This provides a realistic base for evaluating control performance. The model focuses on shuttle-level control within a single system level, excluding vertical lifts and warehouse-wide components.

Evaluation and Analysis
Experiments are conducted to evaluate how each control architecture performs under dynamic operating conditions. Performance is assessed using a targeted set of Key Performance Indicators (KPIs), including throughput, efficiency, scalability, and robustness. Simulation results are compared across architectures to determine the architectural impact on system-level performance and to assess the viability of distributed control principles in future SBSRS design.

1.5. Report Structure

This report is structured to provide a logical progression from the analysis of SBSRS to the design, implementation, and evaluation of I4.0-inspired control architectures.

Chapter 1 Introduction defines the research motivation, objectives and research questions, and positions the study within the context of emerging I4.0 trends in warehousing.
Chapter 2 Shuttle-Based Storage and Retrieval Analysis offers a general analysis of SBSRS, describing system operations, limitations and improvement needs, while identifying key knowledge gaps. The analysis is based on existing literature to establish a broad system-level understanding.

Chapter 3 Multi-Agent System Control Architectures for SBSRS: A Literature Review investigates applicable control architecture methods from the MAS domain, critically comparing centralized, hierarchical, distributed and hybrid approaches to establish their suitability for addressing SBSRS challenges.
Chapter 4 Conceptual Design of Hierarchical and Hybrid Control Architectures presents two generic control architecture designs, hierarchical and hybrid using MAS. It discusses their agent structures, decision logic, communication flows and coordination mechanisms.
Chapter 5 Case Study Analysis: Vanderlande ADAPTO System introduces the ADAPTO system as a representative industrial use case, conducting a current state performance analysis and aligning its system challenges with those identified in the general system analysis.
Chapter 6 Modelling Approach provides a structured justification for the selected simulation methodology, arguing for a hybrid DES and ABM approach to effectively model both centralized and agent-based control dynamics.
Chapter 7 Simulation Model Development describes (1) the implementation of the hybrid DES-ABM simulation model and the architecture-specific control logic used to test and compare both control architectures. (2) The verification procedures performed to ensure logical consistency, correctness and robustness of the simulation model, while explaining the scope limitations regarding model validation. (3) Experimental Plan defines the experimental design, including warm-up periods, run lengths and scenario matrix, to enable statistically sound comparison of the two control architectures under varying system scales and shuttle densities.
Chapter 8 Simulation Results & Discussion presents and interprets the experimental outcomes, providing empirical evidence on how hierarchical and hybrid MAS architectures impact SBSRS performance.
Chapter 9 Conclusion summarizes key research findings, reflects on contributions and outlines recommendations for future research.

2

Shuttle-Based Storage and Retrieval System Analysis

In this chapter a system analysis of SBSRS is performed. This analysis aims to create a complete understanding of the current state-of-the-art SBSRS and establishes the problem definition for the remainder of this study. The following sub-research questions will be answered.

Sub-RQ1: What are the improvement needs that future SBSRS must satisfy?

The chapter begins with an explanation of the system's main components, its operational process, its control policies and its control architecture. Through a structured literature review, the architecture and control mechanisms of SBSRS are examined, revealing systemic challenges and research gaps. By analyzing these challenges alongside the opportunities offered by Industry 4.0 principles, the chapter identifies key areas for system performance improvement. This builds the argument for redesigning the control architecture as a potential solution. Finally, five improvement needs are defined which will guide the selection of control architectures in the following chapter 3.

2.1. Shuttle-Based Storage and Retrieval System

The warehouse technology known as Automated Storage and Retrieval Systems (ASRS) uses automatic systems for item storage and retrieval while maintaining high speed and accuracy to reduce human work and boost operational efficiency [67]. Different operational requirements have led to the development of multiple ASRS configurations, an overview of which is shown in Figure 2.1. Among these, Shuttle-Based Storage and Retrieval Systems (SBSRS) bring together the storage capacity of conventional ASRS systems with the benefits of multiple agent systems and flexible layout designs. Autonomous shuttles in SBSRS operate horizontally along fixed rails inside multi-tier racking structures and vertical lifts handle inter-level transportation. The shuttle operating within a specific level performs independent storage and retrieval operations across multiple tiers simultaneously. SBSRS have become more popular as retailers and e-commerce companies seek fast order fulfillment solutions because of rising customer demands [79]. The parallel operation of shuttles across different levels in SBSRS systems enhances retrieval speed while reducing congestion and delivering high throughput [42].

A SBSRS contains connected physical elements along with control components which include shuttles, racking structures, lifts, conveyors and supervisory control systems. According to Roodbergen et al., the fundamental design elements of an SBSRS can be represented as a combination of physical design and control logic as shown in Figure 2.2. The physical elements include system choice and system configuration while the control elements entail control logic like storage assignment, batching, sequencing and dwell point strategies. The system performance measurements act as a connection between these layers to determine the effectiveness of system support for facility material handling operations. Both layers need to be understood because they determine how control-level architectural

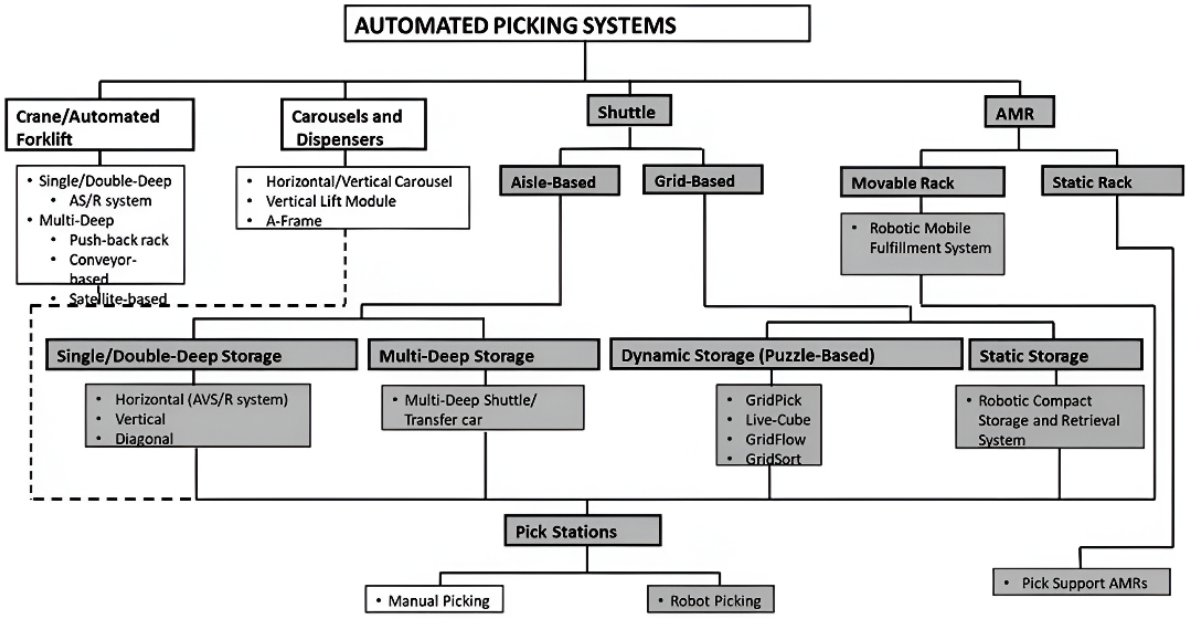


Figure 2.1: Classification of Automated Material Handling Technologies [45].

decisions solve system-level challenges in SBSRS. The physical components and control components of a typical SBSRS are discussed in more detail in the following sections. [67]

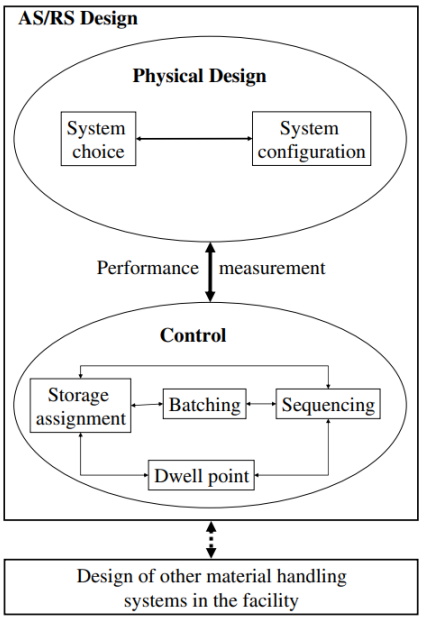


Figure 2.2: Automated Storage and Retrieval System Design Framework [67].

2.2. Physical System Components

The physical composition of a SBSRS is not a one size fits all, it gets modified to fulfill process-specific requirements. The main components include racking structure, lifts, input and output stations, shuttles and Transport Storage Units (TSUs) as can be seen in Figure 2.3. The following sections briefly describe each component that make up a standard SBSRS. [1, 52, 67]

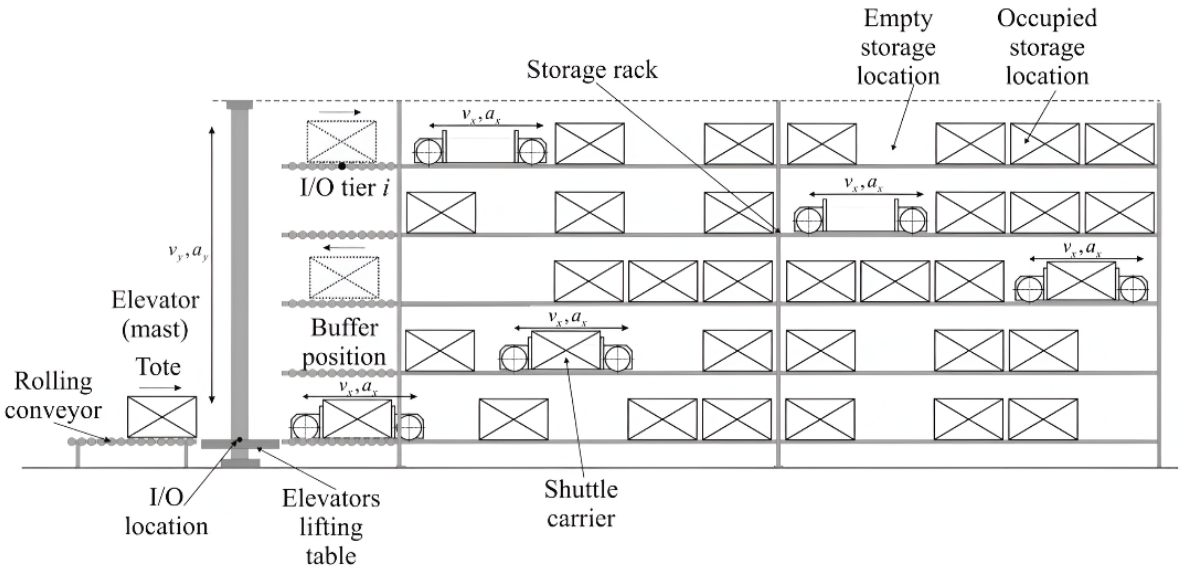


Figure 2.3: Shuttle-Based Storage and Retrieval System Components [52]. A side view of a typical SBSRS.

Racking structure
The SBSRS backbone structure consists of vertical storage lanes or shelves that enable optimized item placement. The system maintains stability through its design which allows shuttles to access stored goods while bearing their weight. The steel racking structure serves two functions: it supports heavy loads of stored items and enables shuttle vehicle travel paths. The racking system adjusts to accommodate various TSU dimensions and customer-specific storage requirements. The structure includes lift shafts together with aisles and optional cross-aisles depending on the SBSRS layout.

Shuttles
The shuttles function as robotized vehicles which move items and TSUs throughout the racking structure. The shuttle uses designated rails to move through the racking structure. The shuttle consists of an electric motor powered wheeled chassis along with batteries for power supply and a load handling system. The load handling device functions to move products between the racking system and the shuttle in both directions. The system uses advanced sensors together with control signals to guide the shuttles. The dimensions and construction of shuttles depend on the weight and nature of the items to be transported.

Lift and lift shaft
The lifts together with their lift shafts function as vertical components which enable the TSUs to move to different levels of the racking structure for storage and retrieval operations [1]. The lifts operate as computer-controlled equipment which transports TSUs either with or without shuttle vehicles, according to SBSRS type, through lift shafts.

Inbound and outbound workstations
The system contains two types of workstations which function as entry and exit points for items before storage or after retrieval. The stations connect to conveyor systems to ensure smooth integration with other warehouse operations. The stations function as entry and exit points for items to access the SBSRS process. The controller receives an initial signal at the inbound station when items enter the system. The outbound station serves as a transfer point which employs conveyors to direct items toward their subsequent logistics operations. The inbound and outbound stations contain conveyor systems with sensors to facilitate item loading and unloading operations.

In-rack pickup & deposit conveyors (buffer position)
The in-rack P&D conveyors function as transfer points and buffers which connect the lift to the aisles [8]. Each level contains these conveyors to connect the lift with the rails and shuttles. Often two or

three positions are available on these conveyors and function to store TSUs until shuttles become available. The storage unit travels from the lift to the pickup conveyor for placement (and queuing) before it gets transferred to a shuttle. The pickup and deposit conveyors maintain buffer positions that enable controlled storage unit movement between the lift and shuttle to support smooth operations.

2.3. Operational Process Flow

An SBSRS carries out operations by handling TSUs through receiving, storing, retrieving and dispatching them. The system optimizes both efficiency and system resource utilization to reach maximum throughput and reduce travel duration. The following sections explains the fundamental material handling processes of a typical SBSRS and thereafter a more detailed explanation is given for the shuttle operations.

2.3.1. Material Handling Process

An SBSRS operates through two primary stages of material handling which include the inbound and outbound procedures. The control systems coordinate all operations by assigning work and managing travel routes and shuttle-lift synchronization.

Inbound Process

The inbound process starts when a TSU reaches the inbound workstation through automated conveyor systems or manual loading points. After the arrival of a TSU, a central controller assigns a storage location through either a Warehouse Management System (WMS) or a Warehouse Control System (WCS). The TSU gets transported to the correct level by the lift. Thereafter it gets transferred to the in-rack P&D conveyor where it potentially has to queue. An available shuttle receives its assignment and picks up the TSU at its pickup point before delivering it to its designated storage location within the racking system.

Outbound Process

The WMS initiates retrieval operations when orders need fulfillment or when the system runs scheduled batch procedures by designating the relevant TSU for retrieval. A shuttle retrieves the TSU from its storage rack before transporting it to the in-rack P&D station for vertical lift transfer. The lift moves the unit to the outbound workstation before handing it over to a conveyor system which continues the process [67].

2.3.2. Shuttle Operations

Shuttle operations in SBSRS are shaped by a combination of physical configuration, control structure, task execution strategies and safety constraints. Together, these factors determine how shuttles transport transportable storage units (TSUs) and how the shuttles perform.

As explained earlier, shuttles are standardized autonomous carriers that execute horizontal TSU transport by moving along rails embedded in the racking structure. The uniform design and capability of shuttles within the system helps simplify maintenance operations and maintain consistent performance across different tiers [3].

Most implementations of SBSRS choose tier-captive configuration which means that each shuttle stays assigned to one particular storage level and cannot move between different tiers. The design choice minimizes route complexity while enabling independent parallel operations across multiple levels. Tier-captivity provides operational simplicity but creates structural limitations for workload distribution between levels since shuttle resources cannot shift between levels to balance demand.

Shuttle movements are generally managed by a centralized or hierarchical control structure. These control structures function through higher-level controllers which perform assignment of tasks, route scheduling and instruction distribution to shuttles while they carry out commands independently from other shuttles [67, 44]. The top-down approach provides a consistent material flow yet it prevents the system from making real-time adjustments in case of congestion or local disturbances.

Beyond the choice of control architecture, shuttle operations are can also be influenced by task execution strategies. One common method is dual-cycle tasking, in which each shuttle completes a combined storage and retrieval operation during a single trip whenever feasible. The strategy cuts down on empty shuttle movements which results in increased system-wide productivity [10]. When dual-cycle opportunities are not available, the shuttle performs storage or retrieval separate from one another based on priority rules or queue management logic.

Finally, safety constraints impose strict movement rules to prevent operational conflicts. Systems implement strict movement restrictions to guarantee safe shuttle operations without any conflicts. A one-shuttle-per-aisle rule often functions as a standard practice for preventing storage aisle collisions [52]. Similarly, cross-aisle access is often strictly controlled, allowing only one shuttle in the cross-aisle at a time which often leads to long waiting times. The necessary constraints help protect system integrity yet they act as performance barriers during periods of high system throughput.

2.4. Control Logic and Policy Overview

The configuration of an SBSRS not only depends on its physical design. It also largely depends on decision-making policies like storage assignment, task allocation and routing. The control strategies are essential for determining system efficiency, throughput, and the possibility for dynamic response. The following sections discuss important control policies used in SBSRS systems found in literature.

2.4.1. Storage Assignment Policies

The storage assignment policy determines an appropriate storage location for a TSU when it arrives. The selection of policies affects both storage and retrieval times and system operational performance. [31, 29, 77, 30]

- Random Storage Assignment: TSUs are randomly stored in available system locations to maintain inventory distribution. This method might result in longer retrieval times.
- Class-Based Storage: The system groups items into demand frequency classes for placing high-turnover items in locations which provide the shortest retrieval distances.
- Dedicated Storage Policy: The system maintains fixed storage locations for each item type which provides straightforward retrieval but results in lower space efficiency.
- Adaptive Storage Policies: The fourth industrial revolution alongside smart warehouses employ AI and real-time data analytics to execute automatic storage location adjustments based on changing demand patterns enhancing system effectiveness.

2.4.2. Task Allocation and Sequencing

Task allocation and sequencing together determine how tasks are distributed across shuttles and in which order they are executed. These decisions strongly affect shuttle workload balance, travel distance, idle time and overall system flow [31, 40, 1, 64]. A wide range of strategies is described in the literature:

- Rule-Based Task Assignment:
  - First-Come-First-Served (FCFS): The scheduling rule operates by performing tasks according to their order of arrival. The implementation of FCFS is straightforward but it fails to recognize optimization possibilities).
  - Shortest Processing Time First (SPT): The system gives priority to tasks that require the least amount of time for completion in order to minimize the total time needed to complete tasks yet may cause shuttle imbalance.
- Dynamic Task Allocation: The most complex scheduling approach involves dynamic control whereby task assignments receive continuous updates based on system conditions which include shuttle availability, queue lengths and congestion levels.
- Dynamic Task Allocation: The most complex scheduling approach involves dynamic control whereby task assignments receive continuous updates based on system conditions which include shuttle availability, queue lengths and congestion levels.

- Auction-Based and Market-Oriented Control: Inspired by MAS, some SBSRS implement auction-based task allocation, where shuttles bid for tasks based on factors such as proximity, battery level and workload. This distributed approach improves adaptability and flexibility.
- Metaheuristic and Learning-Based Methods:
  - Genetic Algorithms (GA): Use heuristic optimization to determine efficient task sequences by considering congestion, task urgency, and shuttle positions.
  - Reinforcement Learning (RL): Allows shuttles to learn effective assignment and sequencing strategies from historical and real-time data. While promising, RL has not yet been widely studied in SBSRS.

2.4.3. Routing and Congestion Management

Shuttle routing strategies define how shuttles move through racking structures to perform their assignments with minimum congestion and maximum efficiency. [31, 55]

- Predefined Routing Strategies: Fixed travel paths and preassigned movement sequences used in predefined routing strategies prevent congestion yet restrict flexibility according to [31].
- Dynamic Path Planning: Real-time path optimization through AI enables adjustments for congestion and shuttle workload and storage/retrieval task priority management.
- Swarm Intelligence and Local Coordination: The use of local coordination through swarm intelligence principles inspired by biological systems has been researched for SBSRS to enable autonomous shuttle coordination for maximum traffic optimization.

2.5. Control Architectures in SBSRS

The control architecture of a SBSRS defines how key decision-making processes, like the ones discussed above, are organized and distributed across system components. Most current SBSRS implementations, both in industry and in research, operate under centralized or hierarchical control architectures [10, 44, 6].

Typically, a top-level Warehouse Management System (WMS) generates task requests, which are dispatched via a subordinate Warehouse Control System (WCS). The WCS maintains global visibility over system resources (shuttles, lifts, storage locations) and centrally assigns tasks and routes to individual devices. Under this architecture, shuttles function as passive executors, following instructions without autonomy or direct peer-to-peer interaction.

This centralized approach offers advantages in terms of global optimization of task sequencing and routing, simplified coordination across the system and proven efficiency under static or low-variability conditions. However, significant architectural limitations have been documented, particularly as SBSRS grow in size and operate under more dynamic environments:

- Scalability bottlenecks: centralized schedulers struggle to compute optimal plans in real time for large fleets of interacting shuttles and lifts [44].
- Responsiveness delays: central control creates latency in reacting to dynamic events such as congestion or equipment faults.
- Robustness risks: the central controller forms a single point of failure [6].
- Limited inter-agent coordination: a lack direct shuttle-to-shuttle communication, inhibiting cooperative behaviors [20].

Existing research on SBSRS control has largely focused on optimizing specific isolated control logic (e.g. scheduling, routing) within that centralized control structure [53, 10]. These methods can boost throughput under stable conditions but remain constrained by the underlying architecture.

Given the limitations of centralized architectures, researchers are exploring distributed control mechanisms and agent-based approaches for SBSRS [58, 44]. In such architectures, decision-making is distributed among multiple autonomous agents like shuttles or zone controllers, which make local decisions based on real-time conditions. This can be implemented in a hybrid form, where a supervisory layer sets high-level objectives while agents self-organize, or as a fully distributed system with peer-to-peer negotiation. Early examples show promise. Kattepur et al. (2018) demonstrated an agent-based

control framework for multi-robot warehouses that achieved near-real-time task allocation and remained operational despite partial failures [44]. Basile et al. (2017) proposed an auction-based mechanism enabling modular fleet expansion and faster task allocation compared to centralized schedulers [6]. While these approaches improve scalability, robustness and responsiveness, they also introduce new challenges in coordination and conflict resolution, requiring carefully designed negotiation and priority rules.

However, such approaches remain rare and largely exploratory in both academic literature and industrial practice. SBSRS literature, in particular, lacks comparative studies evaluating how different control architectures (centralized, hierarchical, distributed, and hybrid) perform under realistic operating conditions. Addressing this gap forms a key motivation for this study. The next chapter is therefore dedicated to exploring control architectures in detail, with a focus on their relevance and applicability to SBSRS.

2.6. The Potential of Industry 4.0

Industry 4.0 (I4.0) describes the transformation known as the fourth industrial revolution which unites advanced state-of-the-art technology with industrial operations. The adoption of I4.0 technologies led to the development of ”smart warehouses” or Warehouse 4.0 by integrating connectivity, data and automation creating flexible and efficient operational systems [54]. Warehouse automation undergoes transformation through I4.0 by implementing several technological pillars including Internet of Things (IoT), Cyber-Physical Systems (CPS) and Artificial Intelligence (AI), as can be seen in Figure 2.4. Together they enable connectivity, digitization and they enable the shift from static automation to dynamic, self-optimizing warehouse environments.

Figure 2.4: Industry 4.0 Pillars [32].

Among these technological pillars depicted in Figure 2.4, several play a direct role in enabling smart warehouse functionality. IoT devices, like RFID tags sensors and smart controllers, perform real-time identification and tracking of products and assets. This produces better visibility into inventory and equipment status. CPS is defined by the integration of computational systems with physical processes. The physical movements of sensor-equipped shuttles and conveyors operate in synchronization with digital data models within a warehouse setting. The digital twin concept enables real-time operation monitoring through virtual warehouse system replicas that support predictive analytics

and what-if decision-making processes [75]. AI and machine learning technologies enhance warehouse automation through intelligent decision-making support like optimizing storage assignments or predicting equipment maintenance. Collectively, these technologies transform a warehouse from a pre-programmed static system into a dynamic data-driven system that self-optimizes and adapts which defines Warehouse 4.0. [75, 54]

One of the foundational frameworks being re-evaluated with the rise of I4.0 is the ISA-95 automation pyramid, shown in Figure 2.5. Traditionally this model structured control across five rigid layers, from sensors and devices at the base to enterprise-level systems at the top [26]. The decision-making process in this model operated through strict upward and downward layer interactions which reduced system responsiveness and flexibility. In contrast, I4.0 replaces traditional vertical systems control hierarchy with horizontal control structures that connect devices and machines directly with software agents across multiple layers. Real-time communication enables agents like shuttles, lifts and TSUs along with machines and software agents to interact across all layers [26]. This allows machine-to-machine coordination and localized decision-making, enabling the system to react immediately to equipment failures or sudden demand surges without waiting for centralized instructions.

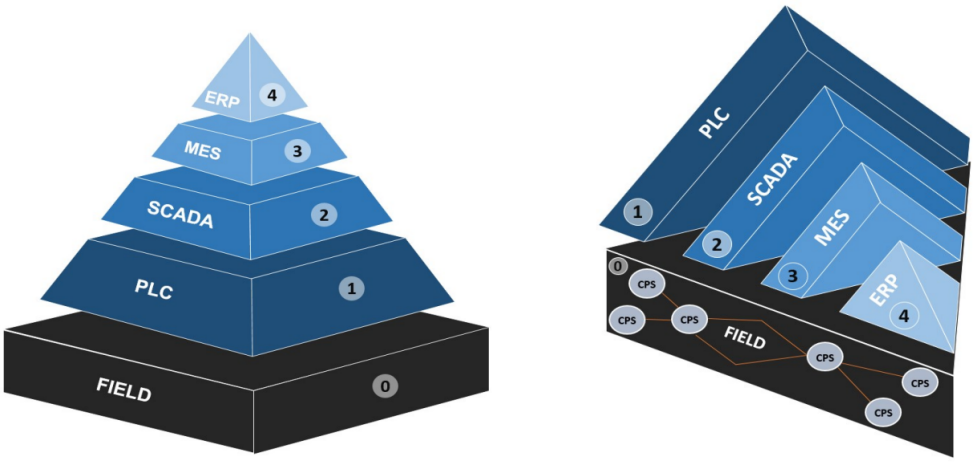


Figure 2.5: Automation Pyramid ISA-95 (left) and Industry 4.0 Architecture with Cyber-Physical Systems (right) [66].

Building on these I4.0 capabilities, recent literature shows an increasing interest in I4.0-enabled warehousing. A framework for flexible and modular warehouse control is proposed by Custodio and Machado, who focus on IoT connectivity and intelligent software agents for scalable and reconfigurable warehouse systems [11]. Their study identifies the inflexibility of the conventional design as a major limitation and argue that adopting distributed control is necessary to ensure alignment with I4.0 principles. They also stress that, to fully exploit I4.0 technologies, system architectures must be reassessed. Not just adding smart devices to old control models but fundamentally shifting toward service-oriented and distributed architectures.

Similarly, the research by Fischer et al., demonstrates how mobile agents enable reconfigurable production systems to perform dynamic task reallocation and system self-organization which further supports agent-based models for flexible intralogistics [24]. These insights are increasingly being adopted, where holonic and multi-agent approaches are experiencing growing interest. The research by Fragapane et al. provides a detailed analysis of autonomous mobile robots (AMRs) in intralogistics while emphasizing sensor technologies and wireless communication and distributed coordination for managing modern distribution center complexity [28]. The research confirms the necessity to shift from traditional static centralized scheduling to adaptive data-driven operations. The ”Warehouse 4.0” model thus requires distributed algorithms for functions like routing and inventory management tasks that are traditionally handled by a central system.

These studies show that I4.0, distributed control and decentralization are increasingly being adopted

in the context of modern warehousing. Although the system is suitable for agent-based control and distributed architectures, the application of I4.0 principles to SBSRS control has not been explored much in the academic literature. This gap demonstrates the necessity for additional research to develop and assess distributed control methods that match the specific characteristics of SBSRS.

2.7. System Challenges and Improvement Needs

The analysis of SBSRS literature shows that operational inefficiencies often stem from control architecture design choices. This section first outlines key systemic challenges that limit current SBSRS performance according to literature. Thereafter five improvement needs are defined for an I4.0-inspired control architecture for future SBSRS. These improvement needs are defined by combining the system challenges and the opportunities offered by I4.0.

2.7.1. System Challenges

This section analyzes key challenges faced by current SBSRS control architectures, based on the literature reviewed. These challenges limit system performance, scalability and adaptability, and reveal critical areas where future improvement is needed. The limitations are categorized and discussed in the following subsections which form the basis for defining the improvement needs in subsection 2.7.2.

Limitations in Traffic and Resource Management

A often documented limitation in SBSRS literature is the restricted movement of shuttles, especially in tier-captive systems where each shuttle is limited to operating on a single level or within one aisle [30, 78]. While this design choice reduces routing complexity, it also makes it harder to balance the workload between tiers and can worsen congestion in certain areas.

Shared resources, such as cross-aisles and transfer points, often become major congestion points. Strict shuttle interaction rules like only allowing one shuttle at a time in an aisle or cross-aisle, lead to queues and leave other parts of the system underused [48, 13]. Because static scheduling policies cannot adjust to changing congestion patterns, these delays often go unmanaged.

Moreover, increasing shuttle density to enhance throughput often results in diminishing returns. Studies show that beyond a certain point, adding more shuttles causes more congestion than benefits, especially in SBSRS with cross-aisles and strict safety constraints [78, 12]. This calls for the need for smarter, traffic-aware coordination methods rather than simply increasing capacity.

To address these issues, future SBSRS control systems should incorporate dynamic traffic control strategies that manage shared resource access based on current system conditions and task priorities. Shuttle routing should adapt to queue lengths and system load, replacing static allocation methods [13]. These improvements would also make it possible to optimize operations at the tier level, preventing gridlock when shuttle density is high.

All of this points to the need for adaptive, dynamic traffic and resource management (IN2) to make better use of shared resources and reduce congestion in future SBSRS.

Limitations of Centralized Control and Scalability

As discussed earlier, most current SBSRS control architectures use a centralized, hierarchical setup where a single controller assigns tasks, coordinates resources, and monitors shuttle movements [38, 24]. While this approach offers complete system oversight, it also comes with notable drawbacks.

As the system grows in size, centralized control becomes less efficient. The number of agents (e.g. shuttles, lifts, and buffers) causes the complexity of real-time task allocation and route planning to grow rapidly, often beyond what can be handled in a timely way [44]. Because all coordination flows through one central point, this setup creates a bottleneck that limits both scalability and responsiveness.

The hierarchical structure also keeps agents strictly separated: shuttles have no direct knowledge of what others are doing. This lack of communication makes it harder to coordinate movements, increases

the risk of local conflicts (for example, in cross-aisles) and prevents cooperative traffic management.

To overcome these challenges, control architectures need to move toward distributed and decentralized designs, where intelligent agents like shuttles can make their own local decisions [44, 24]. Using a MAS approach, agents can choose routes and tasks based on local data, improving scalability, responsiveness and fault tolerance.

These limitations highlight the need for distributed, autonomous decision-making (IN1), modular and scalable system design (IN4) and better inter-agent coordination and communication (IN3) to create systems that are both responsive and resilient.

Structural Limitations in Task Allocation and Adaptive Decision-Making
Most SBSRS in use today still depend on fixed task allocation rules, such as FCFS. These rules don't take into account the current system state, task urgency, or congestion levels. As a result, they often lead to poor task assignments, longer idle times, inefficient routing, and lower throughput when operating conditions change.

Research has shown that methods like reinforcement learning and congestion-aware heuristics can make task allocation more adaptive, prioritizing urgent retrievals and redistributing work to keep the workload balanced [80]. Adding this kind of dynamic task assignment to distributed control architectures can greatly improve system flexibility and efficiency.

In today's hierarchical setups, tasks are assigned without any form of cooperative negotiation between agents. Without dynamic coordination, shuttles can end up taking redundant trips, following conflicting routes and reacting poorly to changes in system demand. To address this, shuttles should have the ability to make local decisions and communicate with each other, allowing them to negotiate tasks and adjust their actions in real time.

This points to a clear need for distributed and autonomous decision-making (IN1), adaptive and dynamic traffic and resource management (IN2), and stronger inter-agent coordination and communication (IN3) to enable real-time, adaptive control.

I4.0 Integration Constraints
Current SBSRS still face limits when it comes to scaling up and adapting to change. As systems grow larger, scheduling tasks becomes more complex, and the lack of local autonomy starts to hurt performance. More decentralization and distributed architectures show a promising alternative because they allow decision-making to happen in parallel and closer to where the work is being done [24].

At the same time, the use of Industry 4.0 technologies in SBSRS is still limited. Adding CPS and IoT infrastructure could greatly improve system responsiveness, self-optimization, and decision-making accuracy [54, 23]. Yet many systems still depend on static, non-adaptive control logic, creating a gap between what's technologically possible and how these systems actually operate.

The next generation of SBSRS should close this gap by building in I4.0 capabilities that support predictive control and real-time adaptability. This could mean using IoT-enabled sensors and communication for live monitoring, digital twins for simulation-based optimization and cyber-physical feedback loops to automatically adjust behavior [49, 23]. These technologies can improve fault detection, reduce unplanned downtime and help systems learn and evolve dynamically.

To make full use of I4.0, future SBSRS should be designed with interoperability and readiness in mind (IN5), ensuring they can seamlessly connect with IoT, CPS, and AI-driven control systems.

2.7.2. Summary of Improvement Needs
Building on the system challenges outlined in section 2.7, it is clear that current SBSRS control architectures face serious limits when it comes to scalability, adaptability, responsiveness, and integration with Industry 4.0 technologies. Studies show that centralized and hierarchical control models often create

performance bottlenecks, especially as systems grow in size and complexity [44, 6]. In addition, fixed routing and rigid task allocation rules make it hard for the system to respond to changing congestion patterns and resource conflicts [13, 53].

Advanced I4.0 technologies offer promising ways to improve SBSRS performance. However, many existing control structures are not designed to work with these technologies. They often lack the flexibility and interoperability needed for future-ready smart SBSRS [11, 28, 85]. To take full advantage of these advancements, the core control capabilities of SBSRS will need to be rethought and redesigned.

From the challenges identified and the findings from the literature review, this research defines a set of improvement needs and system requirements, which are summarized in Table 2.1. The table links each improvement need to the system challenge it addresses most, and these needs will guide the choice of control architectures discussed in the next chapters.

	Improvement Need (IN)	**Rationale (System Challenge)**
IN1	Distributed and autonomous decision-making capabilities	Centralized control creates a single point of failure and limits responsiveness to dynamic events and operational disturbances [44, 6].
IN2	Adaptive and dynamic traffic and resource management	Static routing and rigid resource allocation policies cause congestion and underutilization of system capacity, especially in shared resources such as cross-aisles [77, 13, 78].
IN3	Inter-agent coordination and communication	The hierarchical architecture enforces strict separation between agents, preventing direct coordination and shared situational awareness between shuttles. This structural limitation increases the likelihood of local conflicts (e.g., in shared cross-aisles) and inhibits opportunities for cooperative traffic management and optimized resource utilization. [44, 11, 28].
IN4	Modular and scalable system architecture	Monolithic, centralized architectures do not scale well with increasing system size and complexity, hindering the ability to flexibly expand SBSRS installations [6, 11].
IN5	I4.0 interoperability and readiness	Current control systems are not designed for integration with IoT, CPS, and AI technologies required for smart warehouse capabilities and real-time data-driven optimization [11, 85, 28].

Table 2.1: Improvement needs (IN1-IN5) and corresponding rationale for future SBSRS.

Identifying these improvement needs gives a clear starting point for the architectural choices discussed in the next chapters. In chapter 3 the focus will be on control architectures that can meet these requirements, with particular attention to hybrid and distributed approaches that fit Industry 4.0 principles. Then, in chapter 4, the proposed conceptual control designs will be built directly from these system requirements, ensuring a clear link between the problem analysis and the solutions.

2.8. Research Gaps in SBSRS Literature

Throughout this chapter, several research gaps have been identified. This section brings them together in one overview.

While SBSRS research has made progress in optimizing scheduling and routing, it still lacks a thorough comparison of different control architectures. Most studies focus on improving performance within centralized control systems, without looking at how alternative architectures might affect key system outcomes. The potential of distributed and hybrid control systems to improve SBSRS performance remains largely unexplored.

This gap is becoming more important as I4.0 technologies gain attention for their real-time adaptability and predictive control capabilities. Successfully applying these technologies in SBSRS requires changes to the control architecture that allow for local autonomy, distributed decision-making and continuous data exchange. Integrating them into traditional centralized systems is difficult because these

systems often lack the flexibility and modularity needed to support agent-based, real-time responsiveness.

To properly evaluate SBSRS performance, scalability, and robustness, there is a need for systematic studies comparing centralized, hierarchical, distributed and hybrid control models at the architecture level. Future research should also focus on how to make Industry 4.0 technologies work in practice for SBSRS and on understanding whether the chosen control architecture acts as a driver or a barrier to their effective use.

2.9. Conclusion

This chapter answered sub-research question 1: *What are the improvement needs that future SBSRS must satisfy?* It began with a structural analysis of SBSRS, outlining the main components, operations and control principles. This was followed by a problem-focused assessment of the main challenges in current system designs. The findings from these steps were then combined to define improvement needs, based on both the limitations found and the possibilities offered by Industry 4.0 technologies.

The analysis showed that current SBSRS control architectures do not make full use of what I4.0 can offer. Technologies like IoT and CPS, together with AI-driven analytics, could make systems more adaptable and responsive. However, centralized and hierarchical architectures lack the flexibility needed to integrate these capabilities. Redesigning the control architecture is therefore not just an opportunity to improve performance it is essential for making full use of I4.0 in SBSRS.

From this work, five key improvement needs were defined for future SBSRS, based on challenges found in the literature and the possibilities of I4.0: (IN1) distributed and autonomous decision-making, (IN2) adaptive traffic and resource management, (IN3) enhanced inter-agent coordination, (IN4) modular and scalable architecture, and (IN5) I4.0 readiness. Each of these needs is directly linked to the limitations identified in the system analysis and serves as a clear target for future control solutions.

In addition, this analysis revealed an essential research gap in current academic work. There has been significant progress in optimizing SBSRS operations in the past decades but these efforts have predominantly focused on improving control policies like task scheduling and routing within centralized or hierarchical control frameworks. The architecture itself is rarely questioned. With the growing adoption of I4.0 principles, this is becoming a more pressing issue. There is little research on how hybrid or distributed control architectures could be applied to SBSRS, and even fewer comparative studies on how different architectures affect performance. Without such comparisons it is difficult to know which approaches could best solve existing problems.

All in all, the system challenges discussed in section 2.7 and the knowledge gaps outlined in section 2.8 show the need for alternative control architectures that can meet the improvement needs (IN1-IN5) of future SBSRS. These findings form a clear foundation for the next step in this research: exploring suitable control architectures from the literature. The following chapter will review these architectures, focusing on their ability to meet the improvement needs identified here and their potential for supporting the development of future SBSRS.

To close this gap, the rest of this thesis will explore alternative architectural control approaches, with a particular focus on multi-agent systems (MAS). The next chapter will evaluate different MAS architectures to determine their suitability for meeting the system requirements defined in this chapter. This exploration will be essential for selecting a control design that aligns with the identified improvement needs.

3

Multi-Agent System Control Architectures for SBSRS: A Literature Review

In response to the architectural limitations identified in the system analysis, chapter 2, the aim of this chapter is to identify which control architecture is most suitable for tackling the system challenges and for meeting future SBSRS improvement needs. Doing so, it addresses the sub-research question 2:

Sub-RQ 2: Which control architecture is feasible to meet the system requirements for future SBSRS?

The review looks at four main types of control architectures: centralized, hierarchical, distributed and hybrid. To compare these architectures, the study uses Multi-Agent Systems (MAS) as the main framework for modeling and analysis. MAS offers the flexibility to represent different control setups by changing how authority, communication and decision-making are shared among agents. This makes it well suited for assessing design choices in the context of I4.0 where autonomy, decentralization and integration of cyber-physical systems are becoming more important.

The chapter first introduces the core ideas of MAS, including what agents are, their capabilities, how they communicate, and how they coordinate with each other. It then presents a structured literature review that examines how different MAS-based control architectures affect system behavior in SBSRS. Each architecture type is discussed in detail, outlining its main characteristics, strengths, weaknesses, and how it appears in the SBSRS literature. The insights from this chapter form the theoretical foundation for the design of control architectures in chapter 4.

3.1. Definitions and Key Concepts in Multi-Agent Systems

The following section offers a theoretical basis for MAS. The purpose of this section is to establish a clear understanding of MAS concepts such as definitions of agents, their capabilities, interaction structures and communication mechanisms.

A MAS consists of independent agents which operate in a shared environment to achieve both their individual and collective goals [86]. They are defined as autonomous computational entities able to operate independently in dynamic environments [41]. An agent operates based on its own goals and knowledge, perceives its environment and acts to influence it. The essential characteristics of agents are widely recognized in the literature [86, 41]:

- Autonomy: agents operate independently and control their own actions.
- Reactivity: agents perceive and respond to changes in the environment.
- Proactiveness: agents take goal-directed actions.
- Social ability: agents interact with other agents or humans.

The basic agent-environment interaction loop is shown in Figure 3.1. Agents perceive their environment through sensors before updating their internal knowledge and decision-making models and then act on the environment through actuators to achieve their goals [51]. The perception-decision-action cycle represents a basic behavioral pattern which MAS agents use [86].

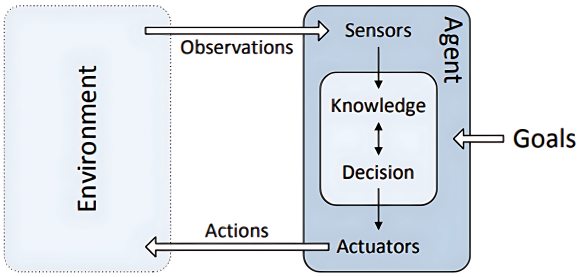


Figure 3.1: Internal Architecture of a Single Agent [51].

Agents need the following core capabilities to function effectively in a MAS according to [49]:

- Sensing: Agents detect important environmental elements which include position and available tasks and system state and local interactions.
- Decision-making: Agents use internal logic and heuristics and negotiation protocols to select actions and plan behaviors while resolving conflicts.
- Movement or action execution: Agents physically or virtually modify the system through movement or operational triggering while following optimized goal-oriented procedures.
- Communication: Exchanging information with other agents, either directly (peer-to-peer) or indirectly (through shared resources or coordination platforms), using mechanisms such as auctions or token protocols [57].

A MAS represents more than just single separate agents, it functions as an organized system which enables agents to work together for system-wide behavior. The basic framework of a MAS is illustrated in Figure 3.2. The system contains agents which both affect the environment and each other while being affected by the environment. The system enables direct agent-to-agent interactions and indirect stigmergy-based interactions through environmental changes. The process of designing a MAS demands both agent behavior definitions and protocols for agent communication and coordination methods [41]. Typical interaction mechanisms between agents include peer-to-peer communication and negotiation protocols and auctions and token-based approaches [57]. The system needs these protocols to reach its desired outcomes by enabling local agent coordination.

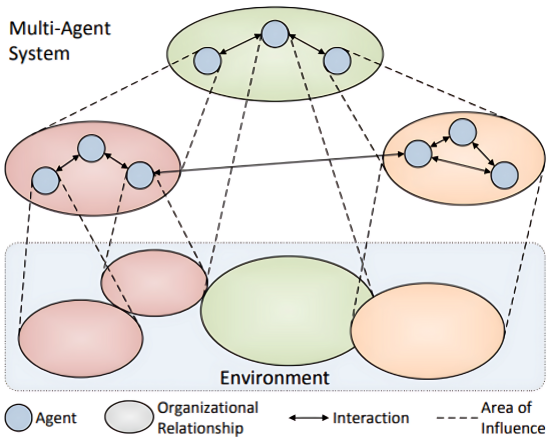


Figure 3.2: A generic structure of multi-agent systems [51].

MAS enable distributed decision-making, cooperation and adaptive behavior, which make them highly suitable for managing complexity in dynamic domains such as manufacturing, logistics and cyber-physical production systems [49]. Next to that, the system becomes more scalable through MAS. Leitão et al. explain that new agents can join without needing system re-engineering [49]. Agents show adaptability because they change their behavior after identifying local changes. Distributed control ensures fault tolerance because it prevents any single failure point from occurring. MAS demonstrate properties which make them a suitable choice for controlling current complex dynamic systems. The following sections evaluate different MAS architectural forms including centralized, hierarchical, distributed and hybrid architectures to determine their capabilities for SBSRS application domains.

MAS are adopted in this study as a modeling paradigm to support the implementation of alternative control architectures for SBSRS. MAS allow the system to be represented as a collection of autonomous, interacting agents that can make local decisions and collaborate to achieve global objectives. MAS is a modeling approach that offers the structural flexibility to implement a range of architectures by varying communication, authority and decision-making structures among agents. This makes it well-suited to design and compare SBSRS control strategies that address all five improvement needs identified in subsection 2.7.2.

3.2. Control Architectures in SBSRS

Control architecture refers to the distribution of decision-making authority and coordination mechanisms within a system. In the context of SBSRS, it defines how key control functions like task assignment, routing and resource allocation, are distributed across system components. The structure of these control interactions directly influences the overall system performance, particularly in terms of responsiveness, scalability and robustness.

Within the field of MAS, the control architecture defines the structure and topology of agent behaviors, their communication patterns and their interaction with the environment [49]. In essence, it specifies who interacts with whom and in what way. Different MAS architectures offer distinct trade-offs between global coordination, agent autonomy, system complexity and adaptability to dynamic environments. Four fundamental MAS control architectures are discussed in this research: centralized, hierarchical, distributed and hybrid control architecture.

A taxonomy is useful for categorizing MAS architectures, but it is just as important to look at how these control approaches are actually used in SBSRS. This section connects MAS theory with SBSRS practice by reviewing how centralized, hierarchical, distributed and hybrid control approaches have been applied, either directly or indirectly, in existing SBSRS research. The review considers not only the formal definitions of each architecture but also what they mean in practice for shuttle-based systems. It focuses on studies that either describe a clear control architecture or use control methods with obvious architectural implications, such as how much autonomy agents have, how communication is organized and how decision-making is centralized or distributed.

This structured review helps clarify how different MAS control approaches affect system behavior in SBSRS. The following subsections analyze each architecture type in detail, highlighting their operational characteristics, benefits, limitations and their application in SBSRS literature.

3.2.1. Centralized Control

Definition, MAS Framing and Characteristics

Centralized control architectures rely on a single decision to manage all operational decisions within the system. The architecture, illustrated in figure Figure 3.3, follows a strict top-down structure in which decision authority is not shared or delegated.

Within a MAS context, a centralized architecture can be seen as a degenerate MAS, in which one central coordinator agent manages and instructs subordinate agents, who execute tasks without autonomy. The central agent maintains complete visibility of system-wide goals and current operational status. It receives updates from sub-agents and resolves conflicts or ambiguities, acting as the sole

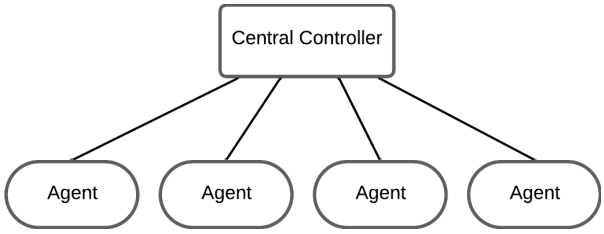


Figure 3.3: Generic Centralized Control Architecture.

decision-maker [50, 69]. Communication in this configuration follows a hub-and-spoke model, with all data funneled to and from the central node. This model is also common in sensor network MAS applications, where a central node aggregates data from distributed sensors and transmits commands based on centralized analysis [10].

Strengths and Limitations

The main strength of centralized architectures is their simplicity and ability to make decisions that are optimized for the whole system. Because the central controller has a complete view of the system, it can use mathematical optimization models, heuristics or scheduling algorithms to reduce duplicated work and ensure everything runs in a coordinated way. For example, Palau et al. describe a centralized prognostics MAS where a single “Social Platform” agent monitors the entire fleet removing the need for agents to communicate directly [69].

However, centralized architectures also suffer from significant drawbacks. The central agent represents a single point of failure, if it malfunctions, the entire system stops functioning correctly. As systems grow in size and complexity, this controller can also become a communication and computation bottleneck which limits scalability [6, 44]. Centralized systems often struggle to respond quickly in real time, since gathering data, making decisions and sending out commands takes time. This delay can make it harder for the system to adapt, especially in fast-changing or congested environments like SBSRS. Another limitation is that centralized control does not take advantage of the parallel decision-making possible in MAS, agents have to wait for instructions which can slow down coordination in high-throughput or large-scale systems. Centralized control still works well in more predictable environments where global coordination and optimization are more important than rapid response or fault tolerance.

Relevance and Application in SBSRS

In SBSRS, a central controller assigns tasks, plans routes and allocates resources, with all system components acting as passive executors. This approach is still the most common in both research and industry. In practice, a WMS or WCS typically manages shuttle and lift operations, deciding on all movements and schedules. Many optimization-based studies use this structure, treating SBSRS control as a global optimization problem that can be solved using heuristics, combinatorial methods, or metaheuristics.

For example, Carlo and Vis (2012) [10] model a tier-captive SBSRS and show that centrally scheduling lift operations can significantly improve throughput by reducing bottlenecks. Wang et al. (2015) [82] use a genetic algorithm to assign shuttle tasks in multi-tier systems, while Yang et al. (2015) [88] combine storage assignment with retrieval sequencing in a centralized setup. More recently, Ekren et al. (2023) [17] also use a top-down control approach to synchronize shuttle and lift operations. These studies achieve strong results in static or moderately dynamic environments but keep shuttles as passive components that simply follow central instructions. As noted by Kattepur et al. (2018) and Basile et al. (2017), this becomes a problem in larger, more dynamic systems, where responsiveness, fault tolerance and scalability are harder to maintain [44, 6].

3.2.2. Hierarchical Control

Definition, MAS Framing and Characteristics
Hierarchical control architectures have multiple layers of decision-making, usually arranged in a tree-like structure where higher-level agents manage those below them, as can be seen in Figure 3.4. Unlike centralized systems with a single decision-maker, hierarchical MAS spread control across levels, with each one responsible for a specific subsystem or function. This setup is similar to industrial hierarchies. For example, a warehouse management system (WMS) overseeing a warehouse control system (WCS), which in turn directs shuttle operations.

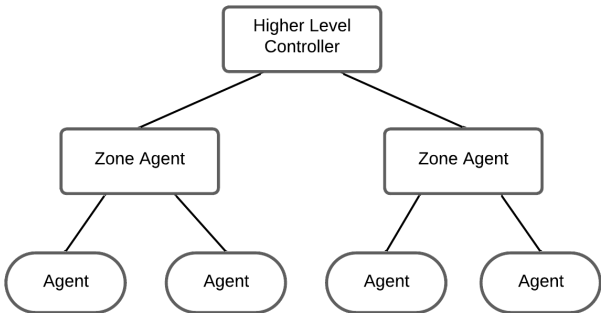


Figure 3.4: Generic Hierarchical Control Architecture.

In a MAS context, agents in a hierarchy work within their own level and manage only their subordinates. Higher-level agents handle broad, strategic tasks, while lower-level agents focus on local operations. Communication mainly flows vertically: subordinates send status updates upward and receive instructions downward [69]. Traditional hierarchies do not allow agents on the same level to communicate directly, although some modern versions permit limited peer-to-peer interaction to improve efficiency and resilience. This layered approach provides partial autonomy in local branches while keeping the system coordinated as a whole.

Strengths and Limitations
The main strength of hierarchical MAS is better scalability compared to centralized control. By spreading authority across different layers, the system avoids a single point of computational overload and can break complex problems into smaller, more manageable parts. Higher-level agents focus on overall coordination, while lower-level agents make operational decisions within set limits. This structure also makes subsystem management clearer and provides defined paths for resolving conflicts [81, 56].

However, hierarchies also share some weaknesses with centralized systems. Strict top-down control can slow responsiveness, as agents may need to wait for approval before acting, making it harder to adapt quickly to changes. Bottlenecks can also appear at intermediate levels if too many sub-agents depend on one supervisor. System resilience is another concern, if a high-level agent fails, an entire branch of the system can stop working, although other branches may continue running. Hierarchies can also be rigid, with fixed command chains that limit local adaptation unless some decentralized elements or “bypass” mechanisms are added [90, 50].

Relevance and Application in SBSRS
Hierarchical control is widely used in SBSRS as a way to manage complexity in large systems. In these setups, a top-level scheduler handles global coordination, like deciding task priorities or balancing throughput, while lower-level controllers manage shuttles, lifts and buffer areas within their zones. This allows tasks to be broken down and handled locally, improving scalability and making the system easier to maintain compared to fully centralized control.

Wang et al. (2016) [81] use a hierarchical queuing approach where shuttles and lifts are treated as separate servers under a central dispatch policy. Cheng et al. (2023) [56] propose a layered scheduling method that synchronizes shuttle and lift movements within a parallel control hierarchy. These designs are similar to real-world SBSRS like the ADAPTO system where tier-captive shuttles are managed by mid-level zone controllers under a central scheduling unit. While this improves scalability, it still relies on a top-down structure. Subsystems can coordinate internally, but individual components remain passive and the architecture struggles to adapt to fast-changing conditions or competition for shared resources.

3.2.3. Distributed Control

Definition, MAS Framing and Characteristics

Distributed control architectures, illustrated in Figure 3.5, remove any single point of authority. Instead, each agent works independently, making decisions based on what it senses locally and on information from nearby agents. In a MAS context, decision-making power is shared equally across agents enabling peer-to-peer coordination without a top-down controller. All agents operate at the same level and the overall system behavior emerges from local interactions guided by shared rules, protocols or negotiation methods [57].

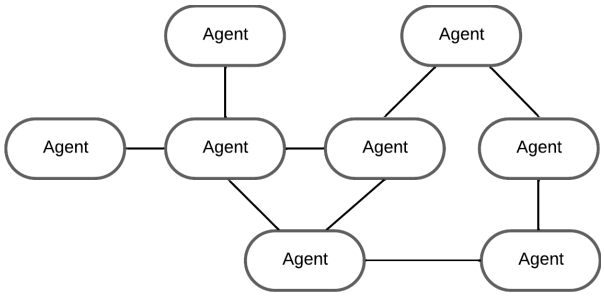


Figure 3.5: Generic Distributed Control Architecture.

Communication in a distributed MAS follows a network-like structure, where agents exchange information directly rather than sending it through a central hub. Decisions are made locally, often using consensus algorithms, voting, or indirect coordination methods such as stigmergy. This approach is common in fields like swarm robotics, smart grids and distributed sensor networks, where quick, local responses are more important than global oversight [57, 7].

Strengths and Limitations

Distributed architectures have major advantages in robustness, adaptability and scalability. Without a central controller, the system can better handle failures, if one agent fails, the others can keep working, and responsibilities can be reassigned on the fly. Decisions can be made in parallel, allowing fast local reactions to changes or disruptions. This makes the approach well suited to unpredictable or fast-changing environments. Examples include drone swarms that reorganize mid-flight or self-healing power grids where local agents reroute electricity when they detect faults [44, 54].

However, the lack of centralized oversight also brings challenges. Without a full system view, agents may choose actions that are locally good but not optimal for the system as a whole, leading to inefficiencies or duplicated work. Coordination is more complex and often needs advanced distributed algorithms to keep the system running smoothly. Other difficulties include extra communication overhead, resolving conflicts and avoiding instability or deadlocks if peer-to-peer interactions are not well managed [84, 54]. Emergent behavior can be unpredictable and ensuring safety or convergence at the system level requires careful design. While distributed MAS may lose some theoretical efficiency compared to centralized systems, their faster responses and fault tolerance often make them a better choice for large or highly dynamic settings.

Relevance and Application in SBSRS

Fully distributed SBSRS designs are rare, but examples from related logistics systems offer useful insights. In these cases, shuttles act as autonomous agents that can choose their own tasks and coordinate directly with each other. For example, Turhanlar et al. (2022) present a distributed control approach for an aisle-changing Automated Mobile Robot (AMR) system, using IoT-based inter-agent communication to prevent collisions and deadlocks [20]. While not developed for SBSRS, this approach is similar to how tier-captive SBSRS could work and shows the potential of agent negotiation without a central controller. However, the study does not cover centralized oversight or measure system-wide performance.

Even with these promising examples, fully distributed SBSRS implementations are still mostly experimental. The main challenges include keeping coordination quality high, avoiding deadlocks and ensuring acceptable overall performance without central planning.

3.2.4. Hybrid Control

Definition, MAS Framing and Characteristics

Hybrid control architectures combine elements of centralized and distributed control to balance global coordination with local autonomy. They can take many forms, one example of which is shown in Figure 3.6. In a MAS context, some agents, usually at higher levels, maintain centralized oversight, while others operate independently or coordinate directly with peers. Rather than strictly following one approach, hybrid systems integrate control principles to match specific operational needs and system layers.

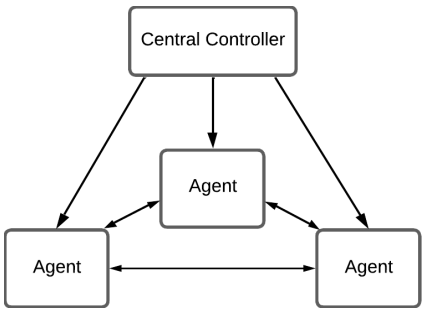


Figure 3.6: Example of a Hybrid Control Architecture.

A common setup is the two-tier “supervisor-worker” model, where high-level agents set overall goals or resolve conflicts, while lower-level agents carry out tasks autonomously under normal conditions. Other designs allow dynamic peer-to-peer interaction within hierarchical layers or include a fallback centralized mode in case of disruptions. This flexibility lets hybrid systems adapt their behavior to changing conditions, improving resilience and responsiveness in dynamic environments.

Strengths and Limitations

The main advantage of hybrid MAS architectures is adaptability. They can keep overall system coherence like optimizing throughput or enforcing fairness, while still enabling fast local decision-making. This helps overcome the scalability limits of purely centralized control and the coordination challenges of fully distributed systems. Some hybrid MAS designs are also adaptive, allowing agent roles and coordination patterns to change during operation.

The downside is that combining different coordination mechanisms makes design and validation more complex. Centralized and distributed modules must work together seamlessly, which requires careful interface design and thorough testing. Modeling hybrid systems analytically is harder than with pure architectures and performance often needs to be validated through extensive simulation or real-world trials, especially for unusual scenarios or rapidly changing conditions.

Relevance and Application in SBSRS

Hybrid architectures are becoming more common in logistics, but in SBSRS they are still relatively rare, despite their strong potential. Ekren et al. (2023) studied a hybrid model for tier-to-tier, aisle-captive SBSRS, where a shuttle and a lift coordinate to synchronize tasks [17]. While this is a valuable contribution, it only covers a small part of what hybrid control can offer and does not extend to a fully hybrid, system-wide design.

Another promising example is from Fan et al. (2025), who integrated real-time multi-agent pathfinding (MAPF) into a hierarchical control structure for a large-scale, grid-based SBSRS. They reported a 31.8% increase in throughput, although the full methodology and results have not yet been made public [21]. It’s worth noting that their system differs fundamentally from tier-captive SBSRS: it allows 3D shuttle movement across a grid, instead of horizontal shuttles working with vertical lifts. While the

results cannot be applied directly to the control challenges in this study, the work is still conceptually relevant as an example of MAS-based decision-making in storage systems.

3.2.5. Key Insights from Literature

A review of seventeen academic studies on SBSRS reveals four main insights into how control architectures are applied. Table 3.1 gives an overview of the most relevant studies, summarizing their architectural focus and main contributions.

First, centralized and hierarchical architectures still dominate in both industry and academia. These systems are defined by strict top-down control, shuttles that act as passive executors, and a focus on global optimization using mathematical scheduling or heuristic algorithms [1, 10, 82, 88].

Second, most SBSRS research aims to improve control policies like task allocation, routing and storage assignment, within an existing architectural setup. Few studies question whether the underlying control architecture itself is the right choice. This mirrors industry's preference for centralized designs, which are predictable and well-established, even though they can be limited in scalability and adaptability [1].

Third, hybrid and distributed architectures are being tested more often in related fields such as autonomous mobile robots (AMRs), automated guided vehicles (AGVs) and UAV fleets. However, such approaches are still rare in SBSRS. Where they do appear, implementations are often small in scope, limited to simulations, or not validated at larger scales.

Fourth, no studies have yet compared different control architectures systematically within the SBSRS context. Even though there is growing recognition of architectural bottlenecks, especially under dynamic, high-density or large-scale conditions, there is still a lack of empirical research on how architectural choices affect key performance metrics like throughput, congestion and responsiveness under I4.0 conditions.

Combining these four insights shows that, while traditional architectures still perform well in static or moderately dynamic settings, early evidence points to hybrid and agent-based designs as having clear advantages in adaptability and scalability. Yet, their potential in SBSRS remains largely unexplored and rarely tested in practice. This leaves a significant research gap: the lack of systematic evaluation and comparison of control architectures under realistic operating conditions. Addressing this gap is a central aim of this study.

3.3. Rationale for Hybrid Architecture

As discussed in chapter 2, most SBSRS today still rely on centralized or hierarchical control. In the same chapter subsection 2.7.2 five key improvement needs (IN1-IN5) were identified that future SBSRS must meet to address these issues. These needs include distributed and autonomous decision-making (IN1), adaptive and dynamic traffic and resource management (IN2), inter-agent coordination and communication (IN3), modular and scalable system architecture (IN4) and I4.0 interoperability and readiness (IN5). The selection of an appropriate control architecture must therefore be guided by its ability to address both the shortcomings identified and the system requirements defined in section 2.7. To assess which architecture is best suited to address the identified improvement needs (IN1-IN5), it is instructive to compare the main alternatives defined in the MAS taxonomy presented in section 3.1.

To support this comparison, Table 3.2 summarizes the key characteristics, strengths and limitations of four main MAS control architectures. As the table illustrates, purely centralized and hierarchical architectures face scalability and responsiveness limitations (IN1, IN2, IN4), while fully distributed architectures lack global coordination and can exhibit unpredictable system-wide behavior, which conflicts with the need for interoperability, optimized resource management and I4.0 integration (IN2, IN3, IN5). In contrast, hybrid control architectures are increasingly recognized in the literature as a promising approach to overcoming the limitations of fully centralized or fully distributed models in SBSRS and related domains [7, 44]. The hybrid architectures combine centralized coordination with local autonomy,

Paper (Year)	SBSRS Type	Control Architecture	Key Control Contribution
Eder (2023) [16]	Aisle-changing SBSRS	Centralized	Analytical queuing model: balancing throughput vs. congestion in multi-aisle systems.
Xu et al. (2016) [87]	Multi-tier, multi-aisle SBSRS with double-deep storage	Centralized	Centralized scheduling for multi-aisle systems improves throughput.
Lerher (2018) *(only summary)*	Tier-captive, multi-aisle system	Centralized	Showed aisle-changing shuttles outperform fixed-aisle systems but require careful centralized scheduling.
Wang et al. (2020)	Multi-tier, multi-aisle SBSRS	Centralized/ Hierarchical	MILP-based system-wide optimization; improved throughput and energy efficiency.
Carlo & Vis (2012) [10]	Tier-captive SBSRS with dual non-passing lifts	Centralized (hierarchical scheduling)	Developed heuristic control to sequence lifts, reducing bottlenecks. Enhanced system throughput.
Wang et al. (2015) [82]	Multi-tier shuttle system	Centralized (optimization-based)	Proposed a genetic algorithm (GA) optimization for task scheduling to maximize throughput and minimize wait times.
Yang et al. (2015) [88]	SBSRS with simultaneous vertical and horizontal movement, aisle-captive	Centralized (hierarchical) control	Integrated optimization of storage assignment and retrieval sequencing. Integer programming + dynamic/heuristic task sequencing.
Ha & Chae (2018) [36]	Tier-to-tier, aisle-captive SBSRS	Centralized (dynamic shuttle allocation)	Developed dynamic ”free-balancing” algorithm to redistribute shuttles across tiers, prevent congestion and improve space utilization.
Wang et al. (2016) [81]	Multi-tier shuttle system	Hierarchical (decomposition)	Conceptualized shuttles/lifts as servers at different stages; hierarchical task decomposition.
Cheng et al. (2023) [56]	Four-way shuttle system, grid-based SBSRS	Hierarchical	Developed queueing-network model; parallel vs. serial lift/shuttle scheduling; hierarchical but no agent autonomy.
Ran et al. (2020) [90]	Multi-aisle, cross-aisle SBSRS	Centralized/ Hierarchical	Dynamic scheduling with event-triggered re-optimization; multi-objective GA.
Ekren et al. (2023) [17]	Tier-to-tier, aisle-captive SBSRS	Hybrid (exact + heuristic)	Hybrid optimization for dual-load lifts; near-optimal real-time task schedules.
Fan et al. (2025) *(only summary)* [21]	Large-scale grid-based SBSRS	Hybrid (hierarchical + MAPF)	Achieved 31.8% throughput gain with real-time MAPF integrated with hierarchical scheduling.
Güller & Hegmanns (2014) [34]	Miniload multi-shuttle SBSRS	Hybrid (Hierarchical/ distributed agent-based)	Multi-agent simulation where each shuttle autonomously selects tasks and routes. Order structure strongly impacts performance.
Turhanlar et al. (2022) [20]	Aisle-changing AMR	Distributed (agent-based + communication)	Fully distributed control with IoT-based inter-shuttle negotiation; real-time deadlock and collision avoidance. Demonstrated flexibility and throughput gains.

Table 3.1: Summary of Key Research on Control Architectures for SBSRS

providing a flexible framework that can effectively meet all five improvement needs of future SBSRS.

Architecture	Key Characteristics	Strengths	Limitations
Centralized	Single central controller manages all decisions; agents are passive	Simple global optimization, unified perspective, straightforward implementation	Single point of failure, poor scalability, limited responsiveness, bottlenecks
Hierarchical	Multi-tier structure with parent-child agent relationships; control flows top-down	Balance between global oversight and local autonomy, layered scalability	Can still experience bottlenecks, limited flexibility in dynamic environments
Distributed	Agents operate autonomously, communicate peer-to-peer; no central authority	High robustness, scalability, fault tolerance, fast local response	Lacks global optimization, high coordination complexity, inconsistent system-wide behavior
Hybrid	Combines centralized coordination with autonomous agents; flexible architecture	Flexible and adaptable, balances global and local optimization, scalable, I4.0 ready	Complex to design, validate and integrate; higher development effort

Table 3.2: Comparison of MAS control architectures for SBSRS.

Hybrid architectures combine the strengths of both approaches: centralized oversight for global objectives and local autonomy for fast, local decision-making. In this setup, high-level controllers handle tasks such as workload balancing, sequencing and resource allocation, while shuttles operate as autonomous agents that make local routing decisions, solve conflicts and adapt to real-time conditions. This division of responsibilities supports IN1-IN3 by allowing responsive, cooperative behavior without constant central intervention. It also improves scalability (IN4), since much of the computational work happens at the local level, reducing the load on the central system as the network grows. Also, hybrid control aligns naturally with I4.0 principles (IN5). Intelligent agents, peer-to-peer communication and real-time data exchange enable seamless integration with IoT, CPS and AI-based tools. Evidence from manufacturing, AGV fleets and warehouse order-picking shows that combining centralized planning with agent-based execution can improve flexibility, responsiveness and robustness [85, 11, 28, 80].

The direct links between the five improvement needs (IN1-IN5) and the core features of hybrid control are summarized in Table 3.3. This overview makes clear how hybrid architectures combine centralized coordination with local autonomy to meet all five needs, providing a scalable, flexible and I4.0-ready solution for the dynamic and complex environments of modern SBSRS.

IN	Improvement Need	Hybrid Architecture Alignment
IN1	Distributed and autonomous decision-making	Local agents (shuttles) make routing and task decisions independently, reducing central dependencies while maintaining global objectives.
IN2	Adaptive and dynamic traffic and resource management	Distributed agents respond to real-time congestion and resource conflicts, while central oversight ensures system-wide balance.
IN3	Inter-agent coordination and communication	Peer-to-peer negotiation protocols allow agents to coordinate directly, reducing delays and improving local conflict resolution.
IN4	Modular and scalable system architecture	Adding shuttles or tiers mainly affects the distributed layer, avoiding central bottlenecks and supporting gradual expansion.
IN5	I4.0 interoperability and readiness	Uses intelligent agents, IoT/CPS integration and real-time data exchange, ensuring compatibility with Industry 4.0 technologies.

Table 3.3: Alignment of hybrid control architecture with the five SBSRS improvement needs (IN1-IN5).

3.4. Conclusion

This chapter explored which control architecture is most feasible to meet the system requirements for future SBSRS, thereby answering sub-research question 2: *Which control architecture is feasible to meet the system requirements for future SBSRS?* To investigate this, the chapter reviewed control architectures within a MAS framework, which allows architectural comparison by flexibly varying the distribution of decision-making, authority and communication among agents [49, 7]. MAS was adopted as the modeling paradigm for this research due to its ability to represent decentralized behavior, support agent-level autonomy, and facilitate coordination in line with I4.0 design principles such as Cyber-Physical integration and IoT-based responsiveness [85, 37].

Based on an in-depth review of MAS theory, architectural taxonomies and state-of-the-art SBSRS literature, a hybrid MAS control architecture emerge as the most promising approach [49]. Hybrid architectures best address the system challenges identified in chapter 2, particularly the improvements needs for distributed decision-making, adaptive resource management, scalability, inter-agent coordination and I4.0 interoperability [21, 17].

A review of 17 studies on SBSRS control architectures highlights four key observations. First, centralized and hierarchical architectures remain the dominant approach in both industrial systems and academic research. Second, these models consistently exhibit top-down control, passive shuttle behavior and global optimization as core traits. Third, although hybrid and distributed control architectures are increasingly explored in related domains such as AGVs, robotic fleets and AMRs, their application to SBSRS remains limited. Fourth, and most critically, no comparative studies currently evaluate the performance of different control architectures in SBSRS, despite the architectural limitations identified in chapter 2.

Recent exploratory work on SBSRS control and related fields shows that hybrid MAS architecture, combining agent-level autonomy with supervisory coordination, can address many of these limitations. These architectures are inherently aligned with I4.0 principles and have shown promise in related logistics contexts for improving throughput, coordination and congestion mitigation [7, 76, 49]. Nevertheless, the literature lacks hybrid MAS designs that are specifically designed for SBSRS and there is a lack of comparative evaluations between control architectures under realistic SBSRS conditions.

To address this research gap, the next chapters will develop, implement and evaluate a conventional hierarchical control model alongside a hybrid I4.0 inspired alternative. Chapter 4 presents the conceptual designs of both architectures, after which simulation-based experiments will compare their performance in representative SBSRS environments. These findings aim to contribute both to academic understanding and to industrial design decisions for future-proof SBSRS control systems.

4

Conceptual Design of Hierarchical and Hybrid Control Architectures

Building on the analysis of control architecture alternatives in chapter 3, this chapter shifts from theory to the conceptual design of practical control models for SBSRS. To compare performance and guide implementation, the architectural principles from the literature are translated into detailed, operational control structures. This chapter addresses sub-research question 3:

Sub-RQ 3: How can a hierarchical and a hybrid control architecture be designed conceptually for a SBSRS?

The goal is to develop two conceptual control models, a hierarchical and a hybrid architecture that meet the system requirements defined in chapter 2. The designs describe the agents' structure and behavior, their decision-making logic and the communication and coordination methods they use. These models will form the basis for the simulation-based evaluation in the next chapters.

By turning architectural theory into conceptual system models this chapter creates a foundation for systematic testing. The resulting designs make it possible to examine how different control approaches influence SBSRS performance under dynamic operating conditions.

4.1. Design Goal and Scope

The goal of this chapter is to present two generic conceptual control models for SBSRS using MAS technology: a basic hierarchical architecture and an Industry 4.0-inspired hybrid architecture. These models are designed to address the main system challenges and improvement needs identified in chapter 2. They also turn the architectural ideas from theory chapter 3 into practical, operational form. The intention is not to model a specific commercial system but to create architecture-independent designs that can be tested in simulations in the following chapters.

This study focuses on the coordination and control of horizontal shuttle movement within a tier-captive SBSRS. The emphasis is on the shuttle agent, its movement, decision-making and communication with other shuttles. As earlier mentioned, since the shuttles operate only on a single level, the scope is limited to horizontal interactions within that level. Vertical lifts and cross-level coordination are excluded.

By focusing only on intra-level control logic (e.g. task allocation, routing and inter-shuttle communication), the models isolate the dynamics of shuttle coordination without the added complexity of vertical transport. While vertical lift control is important for the overall system, including it here would introduce additional layers of resource allocation and synchronization that could obscure the core study of distributed shuttle behavior. Previous research has noted that combining vertical and horizontal control can make it difficult to evaluate local agent behavior in isolation [44].

4.1.1. Design Objectives

The control models in this chapter are designed to meet the five key improvement needs (IN1-IN5) defined in subsection 2.7.2. These needs describe the main functional requirements for future SBSRS control architectures. To turn these high-level needs into practical design choices, we define a set of five design objectives (DO1-DO5). These objectives guide how control mechanisms, agent behaviors and system interactions are built into the models. The design objectives are also shaped by core Industry 4.0 principles, such as decentralization, modularity, adaptability and cyber-physical integration. The following design objectives (DO1-DO5) translate the improvement needs into specific, actionable design decisions:

DO1: Implement local task selection logic at the shuttle level. To meet the need for distributed and autonomous decision-making (IN1) each shuttle selects its own tasks instead of relying on a central controller. Decisions are based on the shuttle's current state, local view of the system and simple utility functions. The logic considers factors such as task priority, travel distance, congestion and system load. This approach improves scalability and allows shuttles to react quickly to changing conditions, reflecting I4.0's principle of distributed intelligence [59].

DO2: Implement inter-shuttle collaboration for shared resource management. To improve coordination and communication (IN3) shuttles can communicate directly with each other to manage shared resources, such as cross-aisles. They share movement intentions, negotiate priority and coordinate timings to avoid collisions. This turns shuttles into cooperative agents capable of resolving local conflicts without constant central oversight, improving overall flow.

DO3: Implement congestion-aware path planning and dynamic rerouting. To support adaptive traffic and resource management (IN2) shuttles use congestion-aware routing. They monitor traffic levels and shared resource availability, adjusting their paths in real time to avoid bottlenecks. This dynamic routing makes the system more responsive and able to self-optimize under changing conditions [85].

DO4: Design modular agent structure supporting scalability. To ensure scalability (IN4) each shuttle operates as an independent agent with its own decision-making logic. New shuttles can be added or removed without major system reconfiguration. This modular structure makes it easier to expand or adapt the system, aligning with I4.0's modularity principle.

DO5: Design for I4.0 integration readiness. While the current models do not implement full I4.0 technologies such as IoT platforms or AI optimization, they are designed to be compatible with them (IN5). Shuttles are conceptualized as cyber–physical entities that can exchange real-time data, connect to external systems and work with technologies such as CPS, IoT and advanced analytics [43].

Together, these five design objectives form the operational foundation for the conceptual control architectures described in the next sections. They ensure the models address the identified improvement needs while preparing SBSRS for future, smart, I4.0-enabled warehousing.

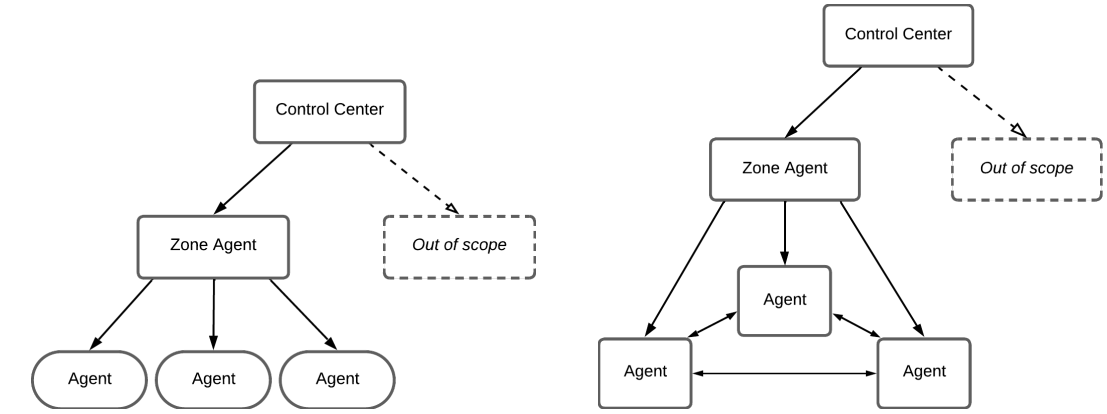
4.2. Control Architecture Overview

This research develops two conceptual control architectures: a hierarchical architecture and a hybrid MAS architecture. These models are designed to put the design objectives defined in subsection 4.1.1 (DO1-DO5), into practice. These models describe how key control functions including task allocation, routing, conflict handling are distributed among system components. By comparing the two, the study evaluates how different control structures affect scalability, responsiveness and coordination in SBSRS.

The **hierarchical control architecture** uses a traditional Warehouse Control System (WCS) design which lets shuttles function as passive executors of commands received from a central controller. The WCS has overall system state awareness to perform task assignments, route selection and operational constraint enforcement. The top-down decision-making method produces optimal results in static or low-variance systems but faces challenges when scaling up operations and managing controller com-

munication and potential controller failure risks [12]. The architecture is visualized in Figure 4.1a where all control decisions flow from the central controller, through zone agents, down to the individual agents (shuttles) which do not communicate directly with each other.

The **hybrid control architecture** implements a semi-distributed system which allows centralized task generation but lets shuttle agents select tasks, determine paths and coordinate activities. The shuttle operates as an independent agent which both detects its environment and makes decisions independently while granting other agents access to shared resources. The architecture maintains global optimization through centralized oversight yet provides local adaptability and system resilience because of agent-based execution. The system design follows cyber-physical systems principles and enables modularity as well as scalability and responsiveness [49, 57]. The control and interaction structure of this architecture is depicted in Figure 4.1b, which illustrates how shuttle agents communicate peer-to-peer in addition to hierarchical flows.



(a) Hierarchical Architecture (b) Hybrid Architecture

Figure 4.1: Conceptual Design of Hierarchical and Hybrid Control Architectures

The differences between the hierarchical and hybrid designs are most visible in their control and information flows. The following sections will examine these differences in detail by studying the control logic, agent behavior and level of decision-making autonomy in the two conceptual models.

4.3. Control Logic and Levels of Decision-Making Autonomy

The following section analyzes how core control functions operate within each architecture. This comparison highlights the key contrasts between hierarchical and hybrid MAS models, especially in terms of decision-making freedom and overall system behavior.

The operational control logic of both hierarchical and hybrid MAS architectures is defined through five core decision-making components: task generation, task selection, routing, conflict handling and communication. How control is shared and how much freedom agents have in each of these areas directly impacts the system's responsiveness, scalability and fit with I4.0 principles. Table 4.1 gives a high-level overview of how these responsibilities are divided between the two architectures. The following subsections look at each control function in more detail.

To compare the two architectures in more detail, the table below breaks down the main control functions side by side. In the hierarchical model, almost everything is centralized, from task generation to conflict handling, with shuttles simply following commands, predefined routes and static rules. In contrast, the hybrid MAS model pushes several decision-making functions down to the agent level. For example, while tasks are still generated centrally, each shuttle selects its next task using a local utility function, which improves distribution and speeds up response times. Routing is no longer fixed but adapts dynamically to local conditions like congestion or shuttle availability. Even conflict resolution is handled locally: instead of waiting for a central decision, shuttles use direct communication and negotiation to manage shared resources such as cross-aisles. This shift gives the hybrid model greater

Control Function	Hierarchical Model	Hybrid Model
Task Generation	Centralized (WMS or WCS)	Centralized (WMS), task pool exposed to agents
Task Selection	Centralized	Local (agent-based utility evaluation)
Routing	Centralized path assignment	Local path finding with dynamic re-planning
Conflict Handling	Centralized prevention and static resolution	Local detection and negotiation-based resolution
Communication	Centralized only	Peer-to-peer messaging and coordination

Table 4.1: Distribution of decision-making responsibilities across architectures.

flexibility, responsiveness and resilience compared to the purely top-down hierarchical approach [57].

4.3.1. Task Generation

The SBSRS generates operational commands through task generation which includes both storage and retrieval requests. The tasks emerge from higher-level business systems like a Warehouse Management Systems (WMS), Enterprise Resource Planning (ERP) platforms or order management systems to represent customer orders and replenishment requirements and incoming goods movements.

The task generation process operates as a centralized function in both control architectures studied in this research. Tasks are not initiated or influenced by the shuttles themselves but are issued by supervisory systems operating at the business-planning layer. The business logic and execution control operate independently from each other which is in line with the standard industrial practice [12, 8].

4.3.2. Task Selection & Task Prioritization

The system uses task selection to choose which shuttle will perform a particular task from the operational requests that are available. Metrics like the shuttle utilization, the system responsiveness to dynamic conditions and its overall system throughput are directly affected by this control logic. Task selection efficiency becomes more important in dynamic SBSRS environments with high density because system state changes quickly.

In the **hierarchical control model** the WCS maintains complete control over task assignment. The WCS uses predefined decision rules such as First-Come, First-Served (FCFS) or nearest-vehicle selection to assign tasks to an idle shuttle from a centralized queue. The shuttle receives no active involvement in task assignment but follows the task plan created by the WCS for execution.

This top-down system design provides simple control but it restricts flexibility when operating within changing dynamic conditions. The centralized approach creates performance challenges through communication delays as well as computational constraints that emerge during real-time state updates to achieve system-wide precision. According to Boysen et al. (2019) rigid assignment methods exhibit slow responses to disturbances and shuttle availability changes which leads to poor resource management [8].

In the **hybrid control model** task selection is delegated to the shuttle agents. The centralized task generator publishes tasks to a common task pool which all agents on their operating tier can access. Shuttles select their tasks from this pool based on local evaluation of current conditions.

A utility function is a common evaluation method in multi-agent task allocation that integrates elements such as [41, 49]: (1) Distance to the task origin; (2) Idle time or availability of the agent and (3) Estimated congestion along the planned route. An example of such a utility function is:

$$U = \frac{\alpha}{d} + \frac{\beta}{a} - \gamma \cdot q$$

where d is distance from the agent to the task origin, a is the agent availability, q is congestion penalty and α, β, γ are tunable weight parameters that determine the relative importance of each factor.

Each agent uses local decision-making to choose tasks based on its specific conditions which leads to improved system responsiveness and workload distribution. Agents that want to execute the same task utilize priority mechanisms such as closest-agent priority or auction-based negotiation for conflict resolution to achieve fair and efficient task allocation [41]. The hybrid MAS model achieves these benefits by moving task selection to the agent level which decreases communication overhead and prevents centralized bottlenecks and enables scalable operation. The model's features match essential I4.0 principles which include modularity and autonomy and cyber-physical integration [49].

Table 4.2 presents the fundamental distinctions between task selection methods across the two architectures by comparing responsibility levels, decision-making criteria, scalability features, response times and communication expenses. The hybrid approach provides better adaptability and scalability to dynamic environments.

Aspect	Hierarchical Model	Hybrid Model
Responsibility	Central WCS assigns tasks to shuttles	Each shuttle autonomously selects a task from shared pool
Decision Criteria	Predefined rules (e.g., FCFS, proximity)	Utility-based evaluation and negotiation
Scalability	Low: centralized bottlenecks under load	High: decisions distributed among agents
Responsiveness	Delayed due to global update cycles	Real-time local adaptation
Communication Overhead	High: requires full system state monitoring	Low: only local or tier-level information needed

Table 4.2: Comparison of task selection strategies between hierarchical and hybrid architecture across various aspects

4.3.3. Routing and Path Planning

Routing is the process a shuttle uses to figure out its path for completing storage or retrieval tasks. This involves deciding which aisle and cross-aisle segments to use, identifying shared access points and planning a route that avoids congestion or obstacles. Good routing is key, it affects how fast a shuttle can move, how well it avoids collisions and ultimately how well the whole system performs. In busy high-density SBSRS environments where conditions change quickly, adaptive routing is especially important to keep operations running smoothly.

The **hierarchical control model** operates with complete centralized routing control. The high level controller assigns tasks to shuttles which then receive their entire route plan during dispatch. The route calculation uses global heuristics to find the shortest distance or minimum completion time. The shuttle has to follow the exact path it is assigned to because a shuttle agent lacks the ability to adjust its route or to make new plans when encountering local disturbances.

The higher level controller implements strict safety policies to make sure conflicts don't take place. System often implement one or more types of safety measures like serialized access to cross-aisles and one-shuttle-per-aisle rules and fixed time windows for shared resources. The safety rules protect against accidents yet they result in underutilized infrastructure and inflexible system operation. The WCS system lacks the ability to forecast situations where two shuttles approach the same cross-aisle from opposite directions which leads to either deadlock or delays. The centralized routing system ensures safety and simplicity yet it limits the ability to adapt and expand in shifting environments.

The **hybrid model** gives shuttle agents the authority to make routing decisions. After choosing a task, agents plan their routes using stored topological maps of the system, represented as weighted graphs. Agents use their local sensing capabilities together with their limited system understanding to detect blocked nodes and estimate congestion while making dynamic route adjustments. The implementation of routing uses A* and Dijkstra's algorithm as heuristic pathfinding algorithms which are

commonly applied in agent-based path planning for flexible material handling systems [55]. The weights of graph edges receive ongoing updates to show both actual and predicted congestion levels, blocked areas and potential deadlock risks. The system handles shared resources like cross-aisles as essential sections. Agents use reservation or time-slot mechanisms to negotiate access to these segments.

For even better coordination, agents should broadcast their planned routes to neighboring peers to prevent conflicts and enable proactive path adjustments. The FCFS access and reservation exchanges protocols (e.g., *broadcast intent* → *claim* → *move*) enable safe and efficient shared resource utilization without requiring central control. The hybrid MAS model enables agents to plan and adapt their routes independently which results in better real-time responsiveness and scalable operation.

The internal design structure of distributed routing in the hybrid MAS model is summarized below. The list outlines key components and their associated decision mechanisms as implemented in the agent-based system design.

- Pathfinding algorithm: A* or Dijkstra's algorithm executed locally by each agent
- Edge weights: Dynamically penalized to reflect congestion, deadlock-prone zones or obstructions
- Conflict zones: Shared areas treated as critical sections with reservation mechanisms
- Access control: Time-slot or soft-locking reservation protocol required before entry into shared zones
- Coordination mechanism: Local negotiation, typically FCFS with intent broadcasting
- Rerouting trigger: Activated upon failed reservation or timeout due to blockage

The local routing method receives strong backing from existing research about agent-based control in manufacturing and logistics which includes PROSA holonic architectures, MAS-based warehouse coordination models and local path planning methods in flexible material handling systems [55, 49, 76]. These researchers show that in dynamic environments, reactive and distributed routing methods perform better than static global optimization methods.

The essential differences between routing methods in the two architectures are shown in Table 4.3 which includes responsibility levels and decision-making criteria and scalability features and response times and communication overhead. The hybrid approach provides better adaptability and scalability in dynamic environments.

Aspect	**Hierarchical Model**	**Hybrid Model**
Responsibility	Central WCS assigns fixed routes to shuttles	Each shuttle autonomously computes and adapts its route
Decision Criteria	Global heuristics (shortest path, minimal time)	Local pathfinding with dynamic congestion adaptation
Scalability	Low: centralized bottlenecks under load	High: routing decisions distributed among agents
Responsiveness	Delayed due to centralized replanning	Real-time local adaptation
Communication Overhead	High: requires full system state monitoring	Low: only local or peer-level information needed

Table 4.3: Comparison of routing approaches across control architectures

4.3.4. Conflict Handling and Deadlock Resolution

The system uses conflict handling and deadlock resolution methods to stop and identify operational conflicts within shared infrastructure segments such as cross-aisles, aisle (intersections) and nodes. These mechanisms are essential for system flow maintenance and safety protection as well as for minimizing prolonged stalls particularly when operating under dense and dynamic SBSRS scenarios.

The **hierarchical control system** operates with centralized conflict management that relies on preventive methods. The Warehouse Control System (WCS) prevents conflicts through path assignment

without overlap and by enforcing strict movement rules. The system follows three standard safety protocols: serialized cross-aisle access, one-shuttle-per-aisle enforcement and FCFS or lowest-ID priority policies.

The WCS system operates with a simple deterministic method that fails to adapt to local congestion or unexpected conflicts. The WCS system does not have real-time access to shuttle interaction developments which prevents it from taking immediate action to resolve blockages. The WCS system implements non-adjustable deadlock prevention methods which reduce system performance and increase response times during spatial conflicts.

The **hybrid control model** allows shuttle agents to perform conflict handling and deadlock resolution duties. The agents predict their planned route and distribute segment reservation data through direct peer messaging. A shuttle agent will determine its next action through local coordination rules by checking signal overlaps from other agents on planned reservations and movement trajectories.

The system controls shared infrastructure elements through reservation-based resource management of intersections and cross-aisles. Agents need to acquire access allowance before they can proceed. The system detects potential deadlocks through repeated reservation failures and shuttle blockages that exceed a specified time period which triggers the agent to perform local resolution methods like safe zone retreats or task urgency-based priority adjustments.

In the hybrid architecture, real-time localized adaptation is possible by integrating conflict detection and deadlock resolution within agents. The agents coordinate autonomously to reduce congestion and to prevent deadlock. The distributed system design minimizes dependence on centralized control systems and enables scalability. The behaviors match the I4.0 principles which include modularity and autonomy and responsiveness [76, 49].

The comparison of conflict handling methods between the two architectures appears in Table 4.4 which examines detection methods along with resolution strategies and system responsiveness and scalability features. The hybrid approach delivers better flexibility and robustness for operating in dynamic SBSRS environments.

Aspect	Hierarchical Model	Hybrid Model
Conflict Detection	Centralized, based on static path assignments	Distributed, based on shared forecast paths and local sensing
Resolution Method	Static priority rules (e.g., FCFS, ID-based)	Local negotiation, rerouting, or safe zone retreat
Deadlock Detection	Absent (rely on avoidance only)	Active, based on failed reservations or cyclic dependencies
Responsiveness	Low. Delayed by central control loop	High. Agents adapt immediately to conflict signals
Scalability	Poor in dense systems due to central bottlenecks	High. Local resolution scales with system size

Table 4.4: Comparison of conflict handling mechanisms in hierarchical and hybrid MAS architectures

4.3.5. Communication

The system allows information transfer between system components by means of communication through vertical paths (shuttle-control systems) and horizontal paths (shuttle agents). The communication system enables fundamental coordination processes that include task selection as well as congestion mitigation and conflict handling and adaptive routing. The systems need effective communication to support local decision-making and system responsiveness in dynamic SBSRS environments.

Under the **hierarchical control model** all communication flows in one direction through centralized

control. The Warehouse Control System (WCS) directs operational commands to shuttles by assigning tasks and providing routing directions as well as timing schedules. The WCS receives status updates and completion acknowledgments from shuttles. The WCS maintains system-wide state information through periodic polling of distributed status reports which serves as the base for all system decisions.

The simplified control logic of this architecture presents multiple challenges for managing dynamic operational settings. Event-driven or periodic communication methods produce delayed responses to new issues that emerge because of congestion and resource conflicts. The shuttle agents lack a direct communication link which prevents them from coordinating their actions or sharing information with each other. The central controller faces an increased risk of bottleneck formation especially when operations occur at high density levels or peak load times because it becomes overloaded.

The **hybrid control model** distributes communication functions to shuttle agents. Agents distribute their planned routes and arrival estimates to peers through peer-to-peer communication while sharing environmental information about blockages, waiting times and negotiating access to common resources. The system becomes more responsive because agents perform autonomous conflict resolution through local communication.

The system uses two types of communication which include direct message protocols like request-grant and yield-deny as well as indirect stigmergic signals through reservation flags placed in the environment for interpretation by other agents.

Table 4.5 evaluates the main communication differences between the two architectures by showing their communication methods, directions, information reach, responsiveness metrics and communication load.

Aspect	**Hierarchical Model**	**Hybrid Model**
Communication Pattern	Centralized only	Peer-to-peer with optional local facilitation
Communication Direction	Unidirectional (WCS → Shuttle)	Bidirectional (Shuttle ↔ Shuttle)
Information Scope	Global system state maintained centrally	Local state awareness and neighbor-level messaging
Responsiveness	Low: response depends on WCS refresh rate	High: event-driven and real-time
Communication Overhead	High: central polling and full-state broadcasting	Low to moderate: localized messaging only

Table 4.5: Comparison of communication structures across control architectures

4.3.6. Summary of Control Logic Components

A SBSRS operates through five essential decision-making elements which include task generation, task selection, routing, conflict handling and communication. The system components establish how it handles dynamic changes while managing resources and adapting to rising complexity levels.

The hierarchical model operates through centralized decision-making for all its operations. The shuttles operate as passive executors without any autonomous decision-making capabilities. The system's implementation becomes simpler while control remains consistent yet its operational flexibility and scalability suffer under different operational conditions.

The hybrid model distributes decision-making authority between shuttle agents. Agents use their environmental perception to make decisions autonomously while they maintain peer-to-peer coordination. The distributed system structure enhances real-time adaptability and congestion handling and system resilience particularly when operating in dynamic or high-density environments.

Control Logic Component	Hierarchical Model	Hybrid Model
I. Task Generation	Centralized WMS/WCS generates tasks; shuttles do not participate	Same centralized generation; tasks broadcast to agent-accessible pool
II. Task Selection	WCS assigns tasks using predefined rules (e.g., FCFS, proximity)	Shuttles autonomously select tasks using utility functions or bidding
III. Routing / Replanning	WCS assigns fixed paths at dispatch; shuttles follow without deviation	Shuttles plan and adapt routes dynamically using local data
IV. Conflict Handling	Centralized avoidance via static rules; no real-time detection	Agents detect and resolve conflicts via local negotiation and rerouting
V. Communication	Unidirectional communication (WCS → Shuttle); no inter-agent interaction	Peer-to-peer messaging and local coordination support distributed control

Table 4.6: Summary of control logic responsibilities in hierarchical and hybrid MAS architectures.

This comparison illustrates the fundamental trade-off between the two architectures:

- The hierarchical control system provides predictable operations with simple design and complete system monitoring yet faces challenges with dynamic environment adaptability and scalability and system responsiveness.
- The hybrid MAS control system achieves flexible and context-aware operations through its ability to grant agents autonomous decision-making power and local interaction capabilities.

The decision-making elements receive implementation and evaluation through simulation to assess architectural performance across different system scales and load levels (chapter 7 and chapter 8).

4.4. Agent Design in the Hybrid Control Architecture

The previous sections have compared the overall structure and control logic of both architectures, but it is important to further clarify how agent-based control is operationalized in the hybrid MAS model. The shuttle system has progressed from being a passive executor to becoming an autonomous agent which combines local decision-making abilities with coordination features. The transformation requires a more detailed internal agent architecture within a single shuttle agent to include perception, reasoning, communication and action capabilities. The following section presents shuttle agent architecture to show how their behavior supports distributed system adaptability.

4.4.1. Shuttle Agent Architecture

The hybrid control architecture implements shuttle design through agent-based system principles to achieve autonomy, reactivity, communication and goal-directed behavior [86, 41]. The architecture consists of four primary components, as discussed in chapter 3: perception, internal knowledge and communication, decision-making and actuation. The internal architectural design is depicted in Figure 4.2 that displays shuttles as systems which maintain continuous environmental interactions.

1. Sensors (Perception Layer): The shuttle agent obtains environmental data by using sensors that track aisle availability, agent position, cross-aisle status and task requests. The system provides real-time observation which enables the agent to understand its environment and adjust its actions.
2. Internal Knowledge and Communication Processing:
    - The knowledge base contains internal state data which includes position information, task queue status, historical interactions and system constraints.
    - The communication module enables MAS-based interaction with other agents. The system enables agents to exchange task bids as well as broadcast congestion signals and negotiate access to shared zones including cross-aisles.

    Together, this enables the agent to perform reactively (responding to environmental stimuli) and proactively (initiating plans based on internal goals and MAS protocols).

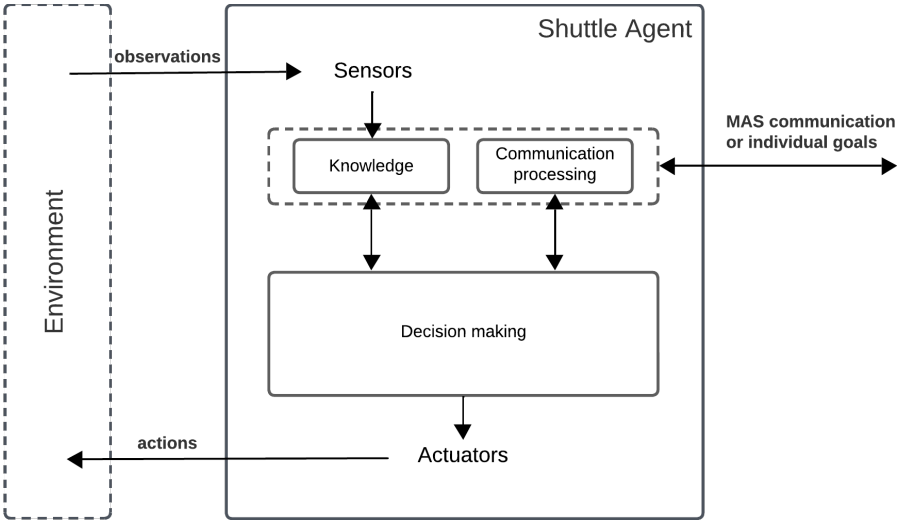


Figure 4.2: Conceptual Design of Internal Architecture of a Shuttle Agent in the Hybrid Control Architecture Model

3. Decision-Making Core: The decision-making module assesses data from the perception layer together with internal knowledge to determine the most suitable action. For example, The shuttle agent has the capability to execute the following operations:
   - The agent has the ability to either accept or reject new tasks based on utility scores
   - The agent can select the most suitable path through the use of local congestion data
   - The agent can either stay at the current location or change route based on the negotiation results. The system operates autonomously without central control which follows the MAS principle of distributed control [49].
4. Actuators (Action Layer): After decision-making the actuators perform the selected physical movements which include storage location navigation, TSU transfer, cross-aisle entry and lift interaction. The final stage completes the control loop which enables the agent to modify its actions through new environmental feedback.

Each shuttle in the hybrid MAS model has gained independent decision capabilities and communication functions which differ from the hierarchical system where they only executed centrally determined actions. Table 4.7 presents key differences between shuttle agent components of these two architectural models. These differences underline the shift from passive to proactive behavior showing how agent-based design enables greater adaptability and resilience in distributed control systems.

Component	**Hierarchical Model (Arch. 1)**	**Hybrid MAS Model (Arch. 2)**
Sensors	Basic sensors to report shuttle position and task status to the WCS	Actively monitor environment (e.g., aisle status, neighboring shuttles)
Knowledge Base	Minimal. The shuttle does not maintain internal goals or task queues.	Stores internal goals, current task queue, local map or heuristics
Communication	Not present for inter-shuttle communication. Only communicates with WCS.	Used to negotiate cross-aisle access, share congestion data, or bid for tasks.
Decision-Making	Fully centralized. Absent. All routing, task allocation and conflict resolution is centralized.	Fully local: decides whether to take a task, how to route and when to wait.
Actuation	Active. The shuttle executes commands received from WCS.	Executes movement based on self-selected plans.

Table 4.7: Comparison of Components of the Internal Architecture of a Shuttle Agent

4.5. Conclusion

This chapter answered sub-research question 3: *How can a hierarchical and a hybrid control architecture be designed conceptually for a SBSRS?* by developing two conceptual control architecture models for SBSRS.

The hierarchical control model functions as a traditional warehouse management system where a central Warehouse Control System (WCS) exercises authority to make decisions about task assignments and routing as well as conflict resolution and system communication. The hybrid MAS architecture distributes intelligence and autonomy across individual shuttle agents which enables localized decision-making, real-time adaptability and peer-to-peer coordination.

The architecture design process was based on five high-level design objectives (DO1-DO5) derived from system improvement needs (IN1-IN5) in chapter 2 and theoretical understandings from chapter 3 MAS literature and I4.0 principles. The proposed models were evaluated through five key objectives: decentralization of decision-making, real-time inter-agent coordination, adaptive task routing, scalability and cyber-physical integration.

The hybrid MAS model demonstrates architectural advantages compared to the basic operational requirements of an SBSRS system. The modular agent-based design enables local task selection through utility heuristics and dynamic routing via local path planning and local conflict handling using reservation and negotiation mechanisms. The communication approach has transitioned from central polling to peer-to-peer coordination thus improving system responsiveness along with robustness without requiring additional central node processing. The combined capabilities solve the main SBSRS problems that stem from centralized bottlenecks and limited scalability and delayed responsiveness.

The hierarchical model maintains its applicability in stable operational environments with stable workloads and predictable tasks. The system's rigid structure provides easy verification and maintains consistency which suits warehouse operations that need strong regulations and safety standards. The system's rigid design hinders its ability to respond promptly and manage localized disturbances effectively which become more significant issues when system complexity and density increase.

The conceptual designs presented in this chapter demonstrate how MAS principles can be applied to SBSRS to support the development of the I4.0-oriented control systems. These models are the foundation on which the simulation experiments that follow are built. These architectural designs will be implemented in a simulation environment through the use of a case study, introduced in the next chapter, chapter 5, to assess their performance across varying operational scenarios. This increases understanding of their practical applicability and comparative strengths.

5

Case Study Analysis: Vanderlande ADAPTO System

Confidential.

6

Modeling Approach

The conceptual control architectures from chapter 4 requires a suitable modeling approach to accurately implement and evaluate these designs. This chapter addresses sub-research question 4:

Sub-RQ 4: Which modeling approach is best suited for accurately implementing the hierarchical and hybrid control architecture?

The modeling approach is selected through a structured method selection process. First, modeling criteria are derived from the characteristics of the conceptual SBSRS model and the specific system requirements defined in chapter 2 and chapter 4. These evaluation criteria assess multiple candidate modeling approaches through insights from both relevant literature and practice. Thereafter reason for using a combined Discrete-Event Simulation (DES) and Agent-Based Modeling (ABM) approach is explained. Finally, the chapter presents the theoretical foundation for the selected modeling approach.

The chapter creates a methodological basis for control architecture implementation and evaluation and thereby bridging the gap between conceptual design and simulation implementation.

6.1. Modeling Criteria

The selection of an appropriate modeling approach is based on the characteristics of the system established in chapter 4. The conceptual models of the SBSRS follow a MAS control architecture. The novel hybrid architecture implements I4.0 principles through distributed control, real-time adaptability and CPS. A computational model requires a specific simulation method which fulfills multiple well-defined criteria to precisely represent system properties. The following criteria are essential for effectively modeling the SBSRS:

1. Distributed Decision-Making: The model must be able to implement distributed decision mechanisms which enable shuttle entities to make decisions independently using both local and shared information.

2. Process Flow Representation: The warehouse operates as a discrete event-driven process because item pickups and storage and retrieval functions happen during particular time points. The simulation needs to accurately simulate these discrete transitions to maintain a structured workflow in the system [47].

3. Multi-Agent Interactions: The agents need to perform efficient system operations through their mutual interactions and coordination abilities. The warehouse environment needs task allocation combined with congestion avoidance and real-time communication to achieve efficient system performance. The simulation requires detailed modeling of these interactions at the granular level. Communication and collaboration between agents should be represented in the model.

4. Scalability: The simulation needs to handle system configurations at different scales because warehouse layouts and operational demands differ based on facility size. The simulation model

should enable scalable system configurations while maintaining efficient computational performance. This scalability is crucial for testing different operating methods and exploring various warehouse layouts under changing workload conditions [46].

5. I4.0 Compatibility: The simulation must be able to integrate CPS and IoT data streams to demonstrate I4.0 capabilities because it allows real-time system monitoring and adaptive decision-making [59].

The five modeling criteria are used to evaluate which modeling approaches from literature best represent the SBSRS models with the goal to compare and evaluate the two control architectures.

6.2. Types of Systems in Modeling

The selection of stochastic versus deterministic models and static versus dynamic models and continuous versus discrete models is essential when modeling a system. These different types of systems are shown in Figure 6.1.

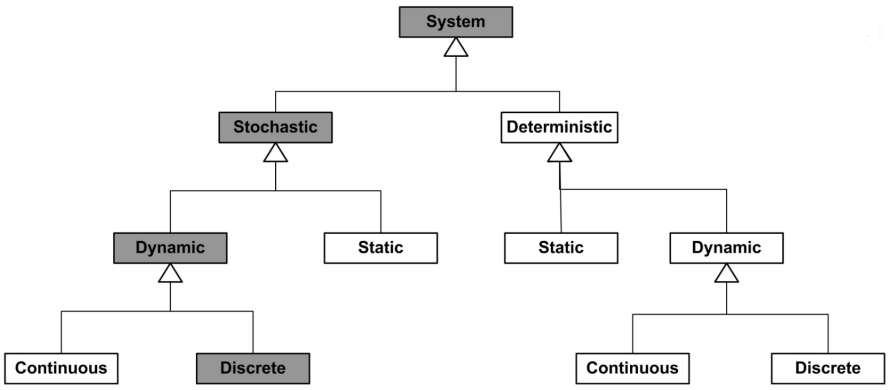


Figure 6.1: Types of Systems in Modeling [68].

Stochastic systems use random variables for modeling unpredictable events thus becoming suitable for modeling systems with variable input and output factors like customer demand as well as processing times and equipment failures. The modeling of deterministic systems produces predictable results which can be expressed through mathematical equations. According to Banks et al. (2005), the stochastic modeling approach should be used when representing an SBSRS because customer demand follows random time patterns [5]. The stochastic nature of customer demand affects storage and retrieval requirements because it produces random variations throughout time. Processing times may also be stochastic due to factors like item condition when stored or retrieved and system congestion.

Systems that transform through time are dynamic, while those that do not change are static. The multiple interconnected components of SBSRS need dynamic modeling because they continuously respond to inputs. This method enables the capture of performance changes that result from demand variations [47]. The system's continuous adaptation to new inputs and conditions makes dynamic modeling more precise than static modeling methods. The operations of SBSRS are affected by changes in product demand which dynamic models enable us to monitor and analyze. The system's nonlinear component interactions require dynamic modeling to simulate their combined effect on total performance.

The timing of events in discrete systems occurs at specific moments in time whereas continuous systems span extensive durations both temporally and spatially. A continuous model describes system behavior through differential or algebraic equations while the system state transforms gradually throughout time. The SBSRS functions as a discrete system because it contains specific activities including storage and retrieval requests as well as unit movements and component interactions [30]. The system operates with discrete entities which include storage units and transportation components.

System behavior is dictated by events like arrival of a storage unit, completion of a storage or retrieval task or error detection.

The decision to use a stochastic, dynamic and discrete model aligns with the complexity and variability of an I4.0-enabled SBSRS.

6.3. Evaluation of Modeling Approaches

6.3.1. Analytical Models vs Simulation Models

Analytical models evaluate key performance indicators through mathematical formulas to determine throughput and cycle time. The deterministic methods in these models produce closed-form solutions according to Gademann (1999) [30]. The initial design optimization phase benefits from analytical models but these models fail to adapt to complex real-time interactions which are common in I4.0 environments where stochastic and dynamic behaviors dominate. The inherent stochasticity, dynamic changes and distributed decision-making in I4.0 systems make analytical approaches inadequate for modeling these systems [2]. Analytical methods like queueing theory and mathematical optimization provide theoretical knowledge yet they fail to represent the dynamic and stochastic warehouse operational characteristics. Simulation models excel at handling stochasticity (random variations) and dynamic interactions which makes them better suited for systems that need adaptability and detailed modeling [5]. Simulation models offer the necessary flexibility to effectively model decentralization features including real-time data analytics and autonomous decision-making. The SBSRS requires simulation-based modeling due to its complex process interactions [47].

6.3.2. Overview of Modeling Methods in Literature

Various modeling techniques exist in the literature for handling different aspects of SBSRS systems and I4.0 contexts. The modeling approaches were assessed for their ability to handle the complex nature of an I4.0-enabled SBSRS system.

- Discrete-Event Simulation (DES): A model uses DES to simulate systems which have events that change at distinct times. It effectively captures high-level operational flows and serves as a great tool for analyzing throughput and 'what-if' scenario [5]. A notable example of DES application is its use in optimizing automated warehouse throughput through the identification of system bottlenecks and performance evaluation under different operational conditions. According to Azadeh et al. (2019), DES has been used to assess operational performance indicators of various robotic storage systems [3]. However, this method fails to handle distributed decision-making processes and adaptive agent behavior modeling.
- Agent-Based Modeling (ABM): Characterizing each part as an autonomous agent, ABM is an approach that fits well when the system to be represented is not centralized and when some emergent behaviors are expected or when reactivity in real time is needed [55]. ABM is suitable for modeling interactions among agents and is an ideal approach for I4.0. ABM has been effectively used in case of warehouses where it has been shown how AGVs can benefit from a distributed control. It lowers the chance of congestion and, in case of a dynamically changing environment, provides a more adaptable response. [55]. ABM has high computation intensity in some cases and can be difficult to validate in others.
- System Dynamics (SD): SD employs a top-down approach that utilizes feedback loops along with stock-flow relationships to analyze system behavior at a high level over time. SD excels at resource management over extended periods and policy evaluation at a high level but fails to deliver detailed system interactions or real-time adaptability for SBSRS systems [72].
- Petri Nets: Petri Nets are a way to model processes both visually and mathematically which helps in understanding workflows and spotting possible deadlocks. They are especially useful for studying how different parts of a system, like shuttles and material flows, work together in sync. In SBSRS, Petri Nets can be used to show concurrent actions and help prevent material handling deadlocks. However, they become harder to use when the system involves complex decision-making or frequent reconfiguration which is often the case in highly dynamic environments. [60, 15]

- Queuing networks: Queuing models describe systems with waiting lines, like service stations and help estimate how much work can be processed, how long delays might be and where bottlenecks occur. In SBSRS, they can give valuable insights into how retrieval tasks flow and where slowdowns happen, especially when arrival patterns change. However, they work best for high-level performance analysis. In situations that require high flexibility and frequent interaction between system components, common in I4.0, they are less effective. [18]

- Digital Twins: Digital Twins create a real-time virtual representation of the physical system, allowing for predictive maintenance and operational optimization [22, 23]. Digital Twins fulfill I4.0 goals yet they require substantial resources to generate precise models of intricate systems. The main challenges include high initial expenses, continuous real-time sensor data integration requirements and maintaining model consistency with the physical system. The digital twin development process requires physical system data access which may not exist and could need supplementary sensors or monitoring tools.

- Hybrid Approaches (DES + ABM): A combination of DES and ABM serves as an effective method for modeling I4.0-enabled SBSRS systems [54]. The high-level operational processes are represented by DES while ABM models distributed agent interactions. The method delivers complete system understanding but is complex and expensive to compute.

Each modeling technique has its strengths and its weaknesses in SBSRS modeling [5, 65, 55]. Table C.1, in Appendix C gives an overview of all the modeling methods found in SBSRS literature together with it (dis)advantages.

6.3.3. Evaluation of Individual Modeling Methods

The requirements set out in Section 6.1 provide the foundation for choosing the most suitable modeling approach for evaluating and comparing hierarchical and hybrid control architectures in SBSRS. The modeling criteria are distributed decision-making, process flow representation, multi-agent interactions, scalability, I4.0 compatibility and implementation feasibility.

The individual modeling methods discussed in the previous section were reviewed and compared against these criteria. The evaluation of each method is given in Table 6.1. A structured rating system assesses their suitability for SBSRS modeling regarding each modeling criteria (++ very strong, + strong, - limited, - - weak).

Modeling Criterion	**DES**	**ABM**	**System Dynamics**	**Petri Nets**	**Queuing Networks**	**Digital Twins**
Distributed Decision-Making	- -	++	- -	- -	-	++
Process Flow Representation	++	-	-	+	++	+
Multi-Agent Interactions	-	++	- -	- -	-	+
Scalability	++	+	++	++	+	- -
I4.0 Compatibility	+	++	- -	- -	-	++
Implementation Feasibility	++	++	+	+	+	- -

Table 6.1: Evaluation of Modeling Methods Against Key Criteria

Discrete-Event Simulation (DES) is widely used in warehouse automation and logistics because it can model system-wide flows and throughput behavior [5, 47]. As shown in Table 6.1, DES scores highly on *Process Flow Representation (++)* and *Implementation Feasibility (++)*, which makes it well-suited for modeling storage and retrieval operations in SBSRS. However, the method performs poorly in *Distributed Decision-Making* and *Multi-Agent Interactions (-)*, meaning it cannot capture the type of

adaptive and autonomous behavior required for I4.0 systems.

Agent-Based Modeling (ABM), on the other hand, performs very strongly in *Distributed Decision-Making* and *Multi-Agent Interactions (++)*, because agents are able to react to local conditions and coordinate with each other [7]. This makes ABM particularly suitable for modeling real-time shuttle coordination and intelligent routing in SBSRS. As indicated in Table 6.1, however, the scalability of ABM is comparatively limited, since simulations with large numbers of interacting agents can become computationally demanding [55].

System Dynamics (SD) achieves high scores in *Scalability (++)* and can be useful for high-level policy and feedback analysis. Nevertheless, its low scores for *Multi-Agent Interactions* and *Process Flow Representation (- -)* illustrate its limited suitability for detailed SBSRS modeling [72].

Analytical approaches, such as Petri Nets and Queuing Networks, perform well in specific areas, for example, process synchronization and queuing behavior but they both score poorly on *Distributed Decision-Making*, *Multi-Agent Interactions* and *I4.0 Compatibility*. For this reason they are more suited for supporting analysis than for full system simulation.

Digital Twins offer very high *I4.0 compatibility* and strong support for *Multi-Agent Interactions (both ++)*, since they allow real-time synchronization between physical and digital systems [22, 23]. However, their *Implementation Feasibility (- -)* is rated as poor due to the high requirements in terms of live sensor data, computing infrastructure and integration effort. As this study does not have access to real operational data, the use of a full Digital Twin is not feasible and is therefore suggested for future work.

Since no single modeling method satisfies all criteria, a hybrid approach is required that combines the strengths of process-oriented control (DES) with the autonomous capabilities of agent-based systems (ABM).

6.3.4. Rationale for the Hybrid DES-ABM Approach

A hybrid DES-ABM framework is the most suitable solution for modeling SBSRS in an Industry 4.0 context because it combines the strengths of both approaches. DES allows the creation of structured process flows, including queuing, storage and retrieval scheduling and is ideal for analyzing system performance and resource use [47]. ABM, on the other hand, enables local decision-making, real-time congestion management and the emergence of self-organizing behavior in autonomous shuttles [55]. By combining both methods the hybrid approach allows simulations that maintain process efficiency while still supporting adaptive, agent-based interactions. This makes the DES-ABM framework particularly useful for scenario testing, congestion analysis and the evaluation of different control strategies, without causing unnecessary computational complexity.

Other hybrid approaches were also considered, but none offer the same balance. For example, combining System Dynamics with DES is useful for high-level policy analysis but not detailed enough to model shuttle-level interactions. Similarly, combining Petri Nets with ABM can model process synchronization, but struggles to represent adaptive decision-making. Integrating a Digital Twin with ABM could provide a very detailed model but would require real-time sensor data and substantial computing resources which are not available in this study. Given these limitations, the DES-ABM hybrid is the most appropriate option, as it provides a realistic, scalable and scientifically sound basis for SBSRS modeling and simulation.

The MAS-based control architectures presented in chapter 4 are built around distributed decision-making, inter-shuttle coordination and real-time adaptability. The DES-ABM framework supports these principles by combining scheduled system events with local shuttle decision processes. As a result it provides a solid foundation for analyzing I4.0-enabled SBSRS performance across different operating conditions and for optimizing control strategies.

6.4. Theoretical Foundation: Discrete-Event Simulation

DES serves as a widely adopted modeling approach for analyzing systems that experience changes through distinct points in time. The method shows its best results when modeling event-driven systems with sequential activities such as queueing, resource allocation and process execution. The system clock advancement in DES occurs only when events take place which makes it a superior method for studying production processes and logistics networks. [5, 47]

The manufacturing and logistics sectors along with healthcare and supply chain management use DES to study system performance through optimization of throughput efficiency, bottleneck identification and system optimization strategy evaluation. Researchers and practitioners can assess different process configurations and control mechanisms before implementation can be done through modeling of task execution, queuing behaviors and resource utilization [48].

DES models often get visualized as process flow diagrams that illustrate event sequences together with entity connections and resource distribution. The diagrams function as blueprints for simulation because they ensure all essential system components and decision rules and dependencies become documented. Process flow analysis helps organizations detect system bottlenecks as well as queue imbalances and resource underutilization by following how entities progress through the system [65].

6.4.1. Core Components of Discrete-Event Simulation

A DES model contains several core elements which determine how a system functions over time. The process-based simulation relies on its fundamental components including entities, events, resources and queues. Each component is shortly discussed in the next sections.

The system in DES modeling exists as a sequence of discrete events which change the system state at each point. Typical system events are arrivals, processing activities, resource allocations and departures that follow established logical rules and predefined distributions. The structured model enables realistic modeling of real-world systems particularly when resource limitations and queue management systems impact operational performance [65]. By tracking system state transitions, DES provides an exact method to study process dynamics and operational efficiency [5]. A DES model defines the system state through the combination of entities, resources and event queues at specific time points. Each event changes the system state because they move entities through space and modify resource quantities and queues.

Entities

The system consists of entities which function as basic units that travel through the system to interact with other components. Entities serve as representations of products, customers, vehicles or tasks based on system modelling needs. Each entity contains attributes which serve to describe its properties like priority levels, process requirements and processing durations. Entity attributes determine how they will interact with resources and waiting queues as well as scheduling mechanisms [47].

Events

The DES system uses an event scheduling approach which executes events at specific times to trigger state transformations. The system maintains an event list which is chronologically ordered to prevent the next event from triggering until the previous event finishes. Common events in DES models include:

- Arrival Events (generator): The system accepts entities through the entrance and allows them to queue up or receive their designated assignments.
- Processing Events: The entity undergoes a defined operation (e.g., manufacturing, assembly or transport).
- Resource Allocation Events: The system allocates resources such as workstations, machines or workers to entities.
- Completion Events: A task reaches its end point and the entity either exits the system or continues to the following process stage.
- Departure Events (sink): The entity leaves the system, completing its journey through the process.

Resources and Allocation Policies
The resources in DES exist as limited system elements that perform operational tasks and handle entities while facilitating system movement. Typical system components are machines, workers and transport units. The availability of resources that entities share among each other directly determines how well a system performs and affects the formation of queues. Resource allocation policies determine the procedures through which entities obtain resource access. Common strategies include:

- First-Come, First-Served (FCFS): Entities are processed in the order they arrive.
- Priority-Based Scheduling: Higher-priority entities receive preferential access to resources.
- Batch Processing: Several entities operate together at once to maximize operational productivity.
- The system distributes resources according to the present operational state of the system.

DES models are often used to assess various resource allocation strategies through bottleneck identification to analyze their effects on waiting periods and congestion levels and operational efficiency [47, 8].

Queues
Queues in DES models store entities while resources become accessible. System performance becomes measurable through queue lengths and waiting times because prolonged delays reveal bottlenecks together with scheduling inefficiencies and resource deficiencies. The system implements queue management approaches such as:

- Single-Server Queues: Entities are processed one at a time.
- Multi-Server Queues: Multiple resources handle incoming entities in parallel.
- Finite Capacity Queues: If a queue reaches its limit incoming entities are blocked or rerouted.

Statistical Distributions
The integration of statistical distributions by DES models allows for the representation of random system variations in process times and arrival rates and resource availability. The implementation of stochastic elements in DES provides both accurate system variability simulation and performance testing and demand fluctuation analysis [5].

All in all, the DES method enables the evaluation of process efficiency and bottlenecks and stochastic variability in complex systems. The modeling capacity of resource dependencies along with queueing behavior and event-driven interactions makes it well-suited for industrial and service-oriented domains. The modeling approach of DES enables efficiency evaluation and strategic planning through entity-based system structure while providing quantitative performance analysis [5].

6.5. Theoretical Foundation: Agent-based Modeling and Simulation

ABM enables autonomous entities known as agents to interact within defined environments through individual behavioral rules and decision-making mechanisms [55]. Each agent in an ABM model possesses defined attributes together with specified behaviors and decision-making rules which determine its actions within a defined environment structure. ABM outperforms traditional DES simulation methods because agents in ABM demonstrate autonomous behavior and adaptability when systems need to respond to changing conditions.

ABM demonstrates its power through modeling complex adaptive systems with distributed decision-making capabilities using autonomous agents. The modeling technique serves to analyze transportation systems and supply chains and finds widespread application in economic and epidemiological fields. The performance of such systems heavily depends on both agent interactions and autonomy [84].

6.5.1. Core Components

A well-structured ABM framework consists of several key components:

- Agents and Their Roles: The system contains autonomous decision-making entities which have specified characteristics including state, location and capability specifications. Agents function autonomously to represent various system elements including people and vehicles and machines and other dynamic system components. According to Macal & North (2010), agents follow deterministic or stochastic decision-making rules that control their behavior [55].
- Environment: In ABM the environment functions as the physical and operational space through which agents interact. The environment establishes limitations that affect how agents behave and enables them to interact with one another. The environment exists either in static form like a warehouse layout or dynamic form like changing market conditions [84]. Through sharing real-time system status data about path availability and congestion levels the environment enables indirect communication.
- Agent Interactions and Communication: Agents potentially exchange data with each other and the environment using communication, competition and cooperation mechanisms. The interactions among agents drive emergent behavior and adaptive decision making [7].

6.5.2. Decision-Making and Emergent Behavior

ABM helps manage distributed decision-making while DES ensures structured event processing. Agents use decision making logic to determine their actions for movement, resource distribution, task completion and other behaviors. ABM decision-making processes can vary from basic rule-based systems to complex adaptive learning systems. Common decision-making logic:

- Rule-Based Decision-Making: Agents follow predefined logic to respond to environmental changes. The method operates efficiently from a computational standpoint yet faces limitations when operating in unpredictable environments according to [84].
- Heuristic-Based Decision-Making: Agents apply heuristics including shortest-path algorithms and priority-based rules to select optimal actions. The approach strikes a balance between operational effectiveness and system responsiveness [7].
- Learning-Based Decision-Making: The agents employ reinforcement learning methods to change their actions based on past events they have experienced. The method provides better flexibility but requires more computational power.

The bottom-up modeling method of ABM produces emergent patterns which result from agent-level activities without needing top-down supervision. ABM proves optimal for studying distributed control and self-organizing systems because of its ability to model complex system-wide patterns from individual agent actions [55].

6.6. Synchronization of ABM with DES

A strong simulation model can be created by combining DES and ABM since this brings together well-structured process flows and local decision-making. DES handles event-driven system dynamics efficiently while ABM allows the simulation to capture emergent behavior from autonomous agents. However, as Tako and Robinson (2012) note, these two approaches must be carefully synchronized to avoid timing inconsistencies and ensure that all interactions remain coherent [73].

The main challenge in hybrid modeling comes from synchronizing different time mechanisms. In DES, time moves forward only when events occur, whereas in ABM agents typically act at fixed time intervals or in response to local conditions [5, 7]. If these two clocks are not aligned properly the system can behave inconsistently and produce unreliable performance results. Another challenge is resource allocation. In DES resources are usually managed centrally while in ABM agents decide independently how to use available resources. This can easily lead to conflicts since multiple agents may try to access the same resource at the same time [54].

To make a hybrid DES-ABM model work properly, the synchronization mechanism has to address both time progression and resource access. A common approach is to use a single simulation clock, controlled by the DES event scheduler which also triggers agent actions. This ensures that both process-level and agent-level events occur in the correct order and at the correct time. Another method is to insert agent updates into the DES event queue as scheduled events so that all actions are handled

within the same sequence.

Resource access is typically handled through a shared interface where agents must request permission from a central (DES-level) resource manager before entering a node or using a path. This allows the DES system to check availability and queue access requests when conflicts occur. Reservation protocols can also be used to prevent future overlap of resource usage. By combining time synchronization and resource coordination the hybrid model maintains both temporal consistency and operational safety, even in highly dynamic environments [55, 54].

6.7. Conclusion

This chapter gives answer to sub-research question 4: *Which modeling approach is best suited for accurately implementing the hierarchical and hybrid control architecture?* The evaluation of simulation methodologies in this chapter revealed that a combination of DES and ABM represents the most effective approach to model SBSRS complex dynamics under various control architectures.

The goal of this chapter was to establish the most suitable modeling approach that would accurately depict and analyze the distributed control of a SBSRS in an I4.0 context. The modeling process followed the criteria specified in section 6.1 which included distributed decision-making, process flow representation, multi-agent interactions, scalability and I4.0 compatibility. The existing literature review showed that no individual modeling approach fulfilled all the modeling criteria. The analysis focused on DES, ABM, System Dynamics (SD), Digital Twins, Petri Nets and Queuing Networks.

The analysis showed that DES is suitable for modeling structured process flows, queuing dynamics and throughput optimization, but it cannot model distributed decision-making and adaptive agent behavior. ABM effectively models autonomous interactions and distributed coordination and emergent behaviors so it is suitable for modeling inter-shuttle communication and real-time adaptability. ABM faces limitations when it comes to structured process modeling and scalability in computational efficiency. Other approaches such as System Dynamics and Petri Nets are useful for macro-level policy analysis and workflow synchronization yet they do not capture the detailed real-time decision-making aspects crucial for an I4.0-enabled SBSRS. Digital Twins were eliminated because they required excessive computational power and real-time data processing and complex system operations.

The research results demonstrated that a DES-ABM hybrid approach offered the best solution. The DES framework enables event-driven modeling of storage and retrieval and queuing operations while ABM enables distributed decision-making and coordination among autonomous shuttles. The combination of DES and ABM allows for simultaneous analysis of system-level efficiency and agent-level adaptability which enables a complete evaluation of operational performance and control strategies and system optimizations.

The methodological foundation established in this chapter supports the next phase of the research, the implementation of conceptual models from chapter 4 in simulation. The hybrid DES-ABM model in chapter 7 will be used to simulate both the hierarchical and the hybrid MAS control architectures for evaluating their effects on SBSRS performance.

7

Simulation Model Development

This chapter presents the development, verification and experimental setup of the simulation model used to evaluate and test the two control architecture designs. The model builds on the modeling principles established in chapter 6 and the use case context from chapter 5 and forms the foundation for the results of the comparative experiments presented in chapter 8. The chapter addresses sub-research question 5:

Sub-RQ 5: How can the hierarchical and hybrid control architecture for SBSRS be implemented and verified in a simulation model?

To give answer to this question the chapter can be divided into three main parts. The first part discusses the modeling of the simulation. Section 7.1 defines the model scope, system boundaries and modeling assumptions to ensure focused and fair evaluation of architectural behavior. Section 7.2 describes the overall simulation structure and hybrid modeling framework, followed by section 7.3 which details the general components of the model that are shared across both architectures, including the layout, shuttle agents, task generation and deadlock handling. Section 7.4 then elaborates on the architecture-specific control implementations, highlighting the differences in decision-making, coordination and routing between the centralized and agent-based designs.

The second and third part of this chapter are respectively the verification of the simulation model and the experimental plan for model comparison. Section 7.5 outlines the verification procedures applied to ensure the internal correctness, logic consistency and robustness of the simulation model. Finally, section 7.6 presents the experimental plan used to evaluate and compare the performance of the two architectures.

7.1. Model Scope & Assumptions

The simulation model aims to evaluate and compare the hierarchical and hybrid control architectural concepts from chapter 4 by implementing them within the ADAPTO use case using a hybrid DES-ABM approach.

As previously mentioned, the focus of this research is on the shuttle agent which means that the simulation model of the ADAPTO SBSRS also focuses on the shuttle agent. In the ADAPTO system the shuttles are tier captive. Thus by focusing on the shuttle agent the scope of the simulation model refines to a single system level, excluding other components like vertical lifts. This allowed for an in-depth examination of shuttle coordination within a controlled operational framework [48].

Next to the shuttle focus, other modeling assumptions and simplifications are applied in the simulation model to maintain focused evaluation and computational manageability.

- Single-level layout: Only horizontal shuttle movement is modeled; vertical lifts are excluded.
- Continuous task supply: Tasks are always available, eliminating arrival-based variability.

- Homogeneous shuttles capabilities: All shuttles have identical speed, capacity and behavior.
- Constant speed movement: No acceleration or deceleration is modeled.
- Positioning delays: Time for alignment of shuttle with the storage location and time for item transfer are included.
- Idealized conditions: No equipment failure and no task execution error aer modeled.
- Steady-state system: Balanced workload by 50/50 chance of storage or retrieval task.
- Uniform item type: All transported units are treated as identical TSUs.

7.2. Simulation Structure and Framework

The simulation model uses a hybrid Discrete-Event Simulation (DES) and Agent-Based Modeling (ABM) approach, as introduced in chapter 6. This combination makes it possible to model both centralized, system-level processes and distributed agent behavior, which is essential for comparing the hierarchical and hybrid MAS control architectures.

To allow a clear comparison, the model is structured so that the general system elements are separated from the architecture-specific control logic. The physical environment, the basic shuttle behavior and the centralized task generation are kept identical in all experiments. The physical system and the agent framework stay the same while the differences between architectures appear in how the core control functions (task assignment, path planning, coordination and safety management) are carried out and by whom. The common simulation components are explained in section 7.3 and the architecture-specific control logic is described in section 7.4. This modular setup ensures that the experimental conditions are consistent and that any performance differences are due to the control architecture itself.

The implementation is developed in Python using:

- SimPy to implement DES-based system processes, including the generation of tasks, shuttle movement and resource control. The simulation timeline advances via a clock-driven event loop using constructs such as `env.process(...)` and `env.timeout(...)`.
- NetworkX to represent the SBSRS layout as a network. Nodes correspond to storage locations, cross-aisles and input/output stations, while edges define permitted travel routes. This graph-based representation supports shortest-path routing, network updates and zone-based movement constraints.
- Custom Python classes to define central agents, shuttle agents, task managers and performance monitors.

The simulation environment integrates DES and ABM into a unified hybrid framework with a clear separation of roles:

- DES Components:
  - *Events:* Task generation, shuttle movement and item handling are triggered as time-stamped events.
  - *Resources:* Access to nodes and zones (cross-aisles, aisles) is controlled using `simpy.Resource` objects. These ensure that shuttles adhere to physical space constraints.
  - *Queues:* Resource contention results in implicit waiting queues. If a shuttle cannot access a node or zone it waits in a queue until the resource becomes available.
- ABM Components:
  - *Agents:* Shuttles are modeled as independent agents. Each maintains internal state variables such as current location, task status, intended path and reservation locks.
  - *Behavior Rules:* Shuttle logic is encapsulated in functions such as `assign_task()`, `move_to_node()` and `perform_task()`, which respectively handle task selection, routing, movement and conflict resolution.
  - *Environment:* The simulation environment (network + SimPy processes) acts as the world in which agents operate and interact.
- Synchronization:
  - Time synchronization is ensured via SimPy's centralized clock (`env.now`) which synchronizes DES events and agent decisions.

  - In the hybrid architecture, coordination between agents is managed through shared resource reservations and limited state sharing (e.g., path visibility to avoid collisions).

This hybrid DES-ABM structure reflects the layered nature of control in SBSRS where high-level decision making and low-level resource interactions happen at the same time. It also allows both control paradigms to be tested using the same simulation infrastructure ensuring that the observed performance differences result from the control logic and not from the modeling method.

7.3. General Model Components

This section describes the fundamental components that form the foundation of the SBSRS model. The structure of the automated system is defined in terms of its physical layout (or environment), agent-based shuttle operations and task management framework. These components remain consistent across both control architectures, with variations in how they are managed and coordinated. The subsequent section 7.4 explains how they are orchestrated differently by the hierarchical and hybrid control logic.

To complement the textual description and to help orient quickly, Appendix D provides visual companions that mirror the control logic discussed. Figures D.1 and Figure D.2 show the shuttle state charts for the hybrid and hierarchical controllers highlighting task execution, CA-Safe recovery and deadlock handling. Figures D.3 and Figure D.4 present the class diagrams detailing the roles and interfaces of the core modules (Shuttle, Controller, SBSRSNetwork) and their data exchange. As you read section 7.3 and section 7.4, refer to these figures to follow the control flow, zone locking and reservation logic more easily.

7.3.1. Environment: Rack and Node Network

The physical layout of the SBSRS is represented as a directed network $G(V, E)$ using NetworkX. The system uses nodes to represent physical locations while edges represent shuttle movement paths between these nodes, as can be seen in Figure 7.1.

- *Nodes V* define the set of all locations in the layout, including:
  - Storage slots `Aisle-j-Slot-k` (orange)
  - Cross-aisle nodes `CrossAisle-i` (green)
  - Input/Output nodes (blue/red)
  - CA-Safe node (purple)
- *Edges E* represent admissible shuttle moves; each has weight = 1 unit travel time.
- *SimPy resources* Every node is a `simpy.Resource(capacity=1)`, ensuring that no two shuttles can occupy the same location simultaneously. The hierarchical model groups entire aisles under one resource with capacity one to study coarse-grained zone-locking due to the constraint one-shuttle-per-aisle.

Figure 7.1 illustrates the network layout used in the simulation. The storage racks exist as longitudinal aisles which contain multiple storage slots within each aisle. The storage locations (orange) within each aisle feature two positions per side of the aisle that use a single-deep storage layout. The cross-aisle (green) allows shuttles to transition between storage aisles while gaining access to all storage positions. The Input/Output points (blue and red) serve as interfaces between the in-rack P&D (Pick-up & Deposit) conveyors and vertical lifts in the physical system. A post-construction check verifies that all storage slots are reachable from both Input and Output; missing edges are auto-repaired and flagged in the log. The model design of the ADAPTO case study functions as a base to implement different SBSRS system sizes and possible system extensions.

To avoid conflicts and ensure safe movement the model uses SimPy resources. Every node, and in some cases groups of nodes such as whole aisles or cross-aisles, is treated as a shared resource. Before a shuttle can move it must first request access to the resource it wants to enter. Depending on the scenario this resource can be a single node or an entire aisle made up of several nodes. This mechanism guarantees that only one single shuttle can occupy one node (or one zone) at a time which

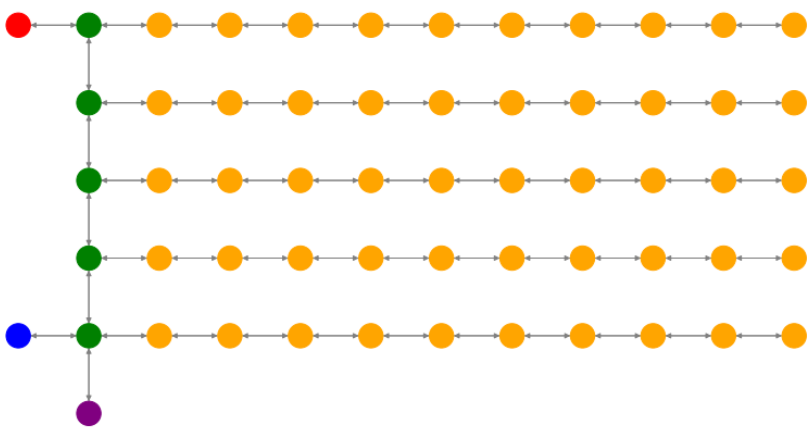


Figure 7.1: ADAPTO layout as a network $G(V, E)$ (I/O points (blue/red nodes); cross-aisles (green nodes); aisle storage locations (orange nodes); CA-Safe (purple)).

prevents two shuttles from entering the same location simultaneously.

This approach to network modeling allows efficient pathfinding, dynamic congestion management, controlled resource distribution and supports modifications to the layout structure for different experimental purposes.

7.3.2. Task Generation

The hierarchical and hybrid models both have centralized task generation from top-level controller agent. The simulation uses separate task delivery methods for each architecture to maintain their respective control logic due to their distinct architectural structures regarding shuttle task assignment.

Every task contains particular attributes which include an item ID, a start location (either the input point or a storage slot) and a target location (either a storage slot or the output point). The target destination for storage operations consists of empty storage slots in the rack but retrieval operations direct to the location where items are stored. The algorithm selects a source location from the occupied storage slots for retrieval tasks and selects a random available storage slot for storage tasks. The simulation maintains consistent item tracking through the generation of distinct item IDs for each new task. The task generation logic is outlined in Algorithm 1.

Algorithm 1 Task Generation Process

Input: Simulation environment, SBSRS network, task controller
Output: Generated storage and retrieval tasks

while simulation is running **do** // Continuous task generation
  Identify occupied storage locations: $stored\_locations$
  Identify available storage slots: $empty\_locations$
  if $stored\_locations$ is not empty **and** random value < 0.5 **then** // Retrieval task (50% probability)
    Select $start\_location$ from $stored\_locations$
    Set $target\_location = Output$
  else // Storage task (50% probability)
    Set $start\_location = Input$
    Select $target\_location$ from $empty\_locations$
  end if
  Generate $item\_id$ // Unique item identifier
  Add task $(start\_location, target\_location, item\_id)$ to ... // Depends on model
end while

Each task $\tau = \langle \text{start}, \text{target}, \text{itemID} \rangle$ is either:

- Storage: start $=$ Input, target $\in$ empty slots
- Retrieval: start $\in$ occupied slots, target $=$ Output

The task generation is the same for both control architectures but how the shuttle handles the task is different for each model. The hierarchical system generates new tasks only after a shuttle becomes idle and are placed into a FIFO task queue. At creation, the slot is marked as *RESERVED* and a unique item ID is assigned. The design prevents storage tasks from referencing occupied slots and retrieval tasks from pointing to already moved items. The central controller uses current system state information to create a valid task which it then assigns directly to the shuttle after receiving a signal from the shuttle about its availability.

The hybrid MAS system employs a centralized task generation process that distributes tasks through a shared task pool. The pool remains active throughout the operation and keeps storage and retrieval tasks in equal numbers at four tasks each. The pool maintains a quantity equal to the four in-rack P&D stations for input operations to match the physical system structure. Shuttles pick tasks from the shared pool through autonomous decision-making using local utility evaluation methods. The system keeps the pool full by adding newly generated tasks to replace the extracted tasks.

7.3.3. Shuttle Agents

Shuttles in the simulation operate as (partially) autonomous agents responsible for transporting items between input/output points and storage locations. Their fundamental behavior and properties remain consistent across all simulation scenarios, ensuring a structured evaluation of different control architectures. This section outlines the key attributes and operational mechanics of shuttles that are common across both model implementations. Variations in control strategies, such as centralized versus local decision-making, are introduced in later sections.

Shuttle Attributes

Each shuttle is instantiated within the simulation environment (SimPy) and has a unique identifier. The following properties define the core attributes of the shuttle agents:

- Current position in the graph $G(V, E)$: At initialization, each shuttle starts at a randomly assigned storage or cross-aisle node, excluding input and output points, to ensure variability in initial conditions.
- Destination (task execution): Shuttles handle both storage and retrieval tasks. A task consists of picking up an item at an assigned location and transporting it to a target destination.
- Task Status: Shuttles track their current state or activity (e.g., idle, traveling, waiting, positioning, item transfer).

Movement Logic & Resource Control Mechanism

Shuttles do not function as resources themselves, they utilize nodes and paths as resources, as explained earlier. Shuttles interact with the environment using SimPy's resource allocation system, which ensures controlled movement, prevents congestion and ensures safety. The SBSRS network is divided in multiple resources, each with a capacity of one shuttle at a time. Depending on the scenario a resource is one node or an entire aisle of nodes. A shuttle must request access to a shared resource before moving into it. If the shared resource is occupied, the shuttle waits until it is available. Once the shuttle moves, it releases the previous node to allow other shuttles to proceed. Next to that, transitioning between aisles, an aisle to cross-aisle transfer or vice versa, incurs an additional time delay to simulate real-world constraints (e.g., wheel reconfiguration or transfer delays). Algorithm 2, in Appendix D, shows what the pseudo-code of the controlled shuttle movement looks like.

Task execution

The execution of a shuttle's tasks follows a state-based structure, transitioning between different operational states such as traveling, waiting, positioning and item handling. Figure 7.2 illustrates the internal statechart that governs the behavior of each shuttle during the simulation. The statechart represents a simple sequence of operational steps and is used to structure the interaction between DES events

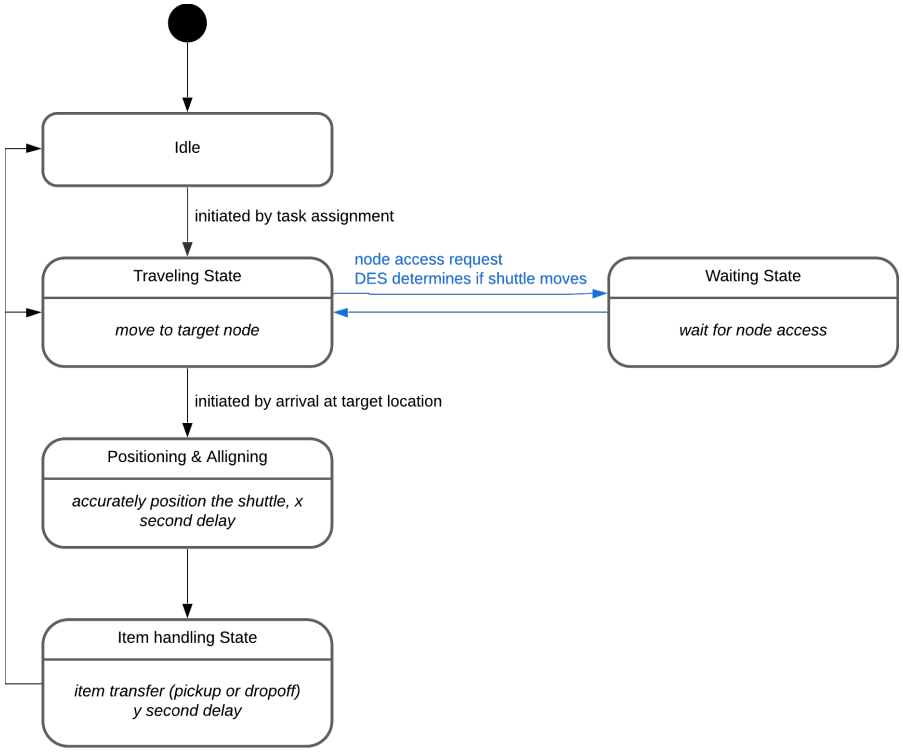


Figure 7.2: A Baseline of the Shuttle State-Chart.

and agent-level decisions.

A shuttle starts in the Idle state and stays there until it receives a task. Once a task is assigned it moves to the Traveling state and heads toward the target node in the network. Before entering each new node, the shuttle must request access. If the node is already occupied, the shuttle enters a Waiting state until it becomes available. Once access is granted it continues traveling. When the shuttle reaches its destination it enters the Positioning and Aligning state, where a short delay is applied to simulate physical alignment at the storage or retrieval location. It then moves into the Item Handling state where the item is either picked up or dropped off. After completing the operation, the shuttle returns to the Idle state and waits for the next task.

This statechart gives each shuttle a clear and consistent behavior pattern and ensures that both control architectures interact with the shuttles in exactly the same way. The difference between the architectures lies not in the shuttle behavior itself but in how and when tasks and access permissions are provided.

Throughout the simulation, each shuttle's location, movement and task status are logged at every time step. This allows for detailed tracking of performance metrics such as shuttle utilization, travel time and waiting time, providing insights into system efficiency across different control architectures.

7.3.4. Deadlock Resolution

The standardized deadlock resolution procedure, detailed in Algorithm 3 in Appendix D, proceeds as follows:

1. Pause current task and mark $is\_resolving\_deadlock = True$.
2. Compute shortest path to CA-Safe and traverse it under normal locking rules.
3. Release any held node locks; set $in\_safe\_zone = True$.
4. When the cross-aisle is clear the shuttle resumes the saved task.

This procedure is implemented identically in both control architectures. The only distinction lies

in how deadlock conditions are detected, either through a centralized monitoring mechanism in the hierarchical model or through local path forecasting in the hybrid MAS approach.

7.4. Architecture-Specific Control Implementations

This section describes the simulation-specific implementation of the two control architectures introduced conceptually in chapter 4: a traditional hierarchical model and a hybrid MAS model. Both architectures operate on a shared simulation backbone, explained in the previous section, but differ fundamentally in decision-making, routing and coordination mechanisms.

7.4.1. Hierarchical Control Implementation

In the hierarchical control architecture all decisions are centralized in a controller. This mirrors conventional SBSRS logic emphasizing system-wide optimization and strict coordination. The execution proceeds as follows:

1. Central task assignment
    - A central controller monitors a FIFO task queue and all shuttle states.
    - Every simulation tick, it checks for idle shuttles using a `SimPy` process (`env.process(self.run())`).
    - When a shuttle becomes idle, a new task is generated and immediately assigned (see Algorithm 1).
2. Global path planning
    - The full route is computed using `networkx.shortest_path()`.
    - The controller locks all nodes along the route using `simpy.Resource` requests (pessimistic locking).
3. Conflict handling and deadlock recovery
    - Exclusive path reservations structurally prevent runtime collisions.
    - If a shuttle is stationary on a cross-aisle for more than $\Delta t = 3$ seconds, the controller reroutes it to a CA-Safe zone (see Algorithm 3).
4. Shuttle execution
    - Shuttles act as passive executors.
    - No local logic or inter-agent communication is employed.

7.4.2. Hybrid MAS Control Implementation

The hybrid MAS architecture decentralizes decisions to the shuttle level, relying on shared data structures and indirect communication to achieve system-level coordination. Key components are as follows:

1. Shared task pool access
    - Storage and retrieval tasks are placed in capped pools (max four tasks each), reflecting system constraints and P&D station capacity.
2. Utility-based local task selection
    - Each idle shuttle computes a utility score U for each task:

$$U = \frac{\alpha}{d} + \beta I + \lambda(1 - e^{-k\alpha}) - \gamma R$$

      where d is the distance to task, I is idle time, a is task age, and R is conflict risk.
    - The task with the highest U is removed atomically from the pool.
3. Path forecasting and conflict avoidance
    - Shuttles forecast their intended paths 4 time units ahead and write $\langle \text{node}, \text{timestep} \rangle$ entries into a global reservation hash map.

4. Rolling lock and conflict resolution

    - Only the next two nodes are locked at any time.
    - Overlapping forecasts are resolved using a yield rule (closest to CA-Safe proceeds).
    - Persistent overlaps ($\Delta t > 3$s) trigger automatic rerouting.

5. Autonomy and adaptiveness

    - Agents independently observe the environment and adjust behavior through local rerouting, delay, or backoff strategies.
    - This supports emergent behavior and distributed coordination.

7.4.3. Summary of Control-Level Differences

To highlight how the two control architectures differ in practice, Table 7.1 summarizes the main implementation differences between the hierarchical and hybrid MAS models. The comparison focuses on the most relevant control aspects, including task allocation, path reservation, conflict handling, communication and agent autonomy.

Table 7.1: Implementation Differences between Hierarchical and Hybrid MAS Architectures

Aspect	**Hierarchical Control**	**Hybrid MAS Control**
Task Allocation	Centralized push by controller	Local pull using utility-based function
Path Commitment	Entire path pre-reserved	Rolling two-node reservation window
Conflict Strategy	Pessimistic exclusive locks	Forecasting + yield + reroute
Communication	One-way (Controller → Shuttle)	Shared reservation table
Agent Autonomy	None (passive execution)	Semi-autonomous shuttles
Scalability Limit	Controller CPU and path queueing	Cross-aisle usage and local contention

These distinctions reflect the architectural trade-offs between rigid central control and distributed agent-based flexibility. Their quantitative impact on performance is explored in chapter 8.

7.5. Simulation Verification

The scientific reliability and meaningfulness of simulation results in this research depend on thorough verification of the simulation model. Verification and validation (V&V) are well-established best practices in simulation modeling and are critical to establish confidence in model implementation and output results [70, 48]. The verification process ensures the simulation model operates as intended by its conceptual design through error detection and logic flaw identification. The validation process confirms that the model accurately represents the real-world system it aims to simulate [70].

The study performs complete verification but it does not include empirical validation of the simulation model. The process of validation demands that simulation outputs get compared to actual data collected from operational systems [70, 48]. Although real-world data from the use case, the ADAPTO system, is available, empirical validation was not pursued as direct comparison would not be meaningful. This study isolates and simplifies shuttle-level logic in a single-tier setup, whereas the actual ADAPTO system includes vertical lifts, complex scheduling and full-system control integration. Nonetheless, the hierarchical control logic implemented in this study reflects the principles used in ADAPTO's shuttle operations and the hybrid control model is grounded in both agent-based literature and system-specific constraints. The hybrid MAS architecture studied in this research is a conceptual future design which has not been implemented in practice in comparable systems. Since the objective is to compare control architectures rather than replicate exact performance, model verification and consistency with known use case control behaviors were considered sufficient.

Because the study's goal is to compare control architectures under controlled conditions, verification of internal correctness was deemed sufficient. It is essential to confirm that any observed performance differences are the result of architectural choices and do not come from modeling errors. The model undergoes thorough verification through modular verification, deterministic testing, flow consistency

checks, spatial constraint verification and robustness testing for edge cases. The verification approach adheres to best practices by following the guidelines established by [48, 25, 70, 65].

7.5.1. Modular Verification

It is crucial to verify the model not only after implementation but also during the process. Therefore, some additional lines are added to the simulation to avoid errors while coding. Initially, bugs have been prevented by adopting a modular design approach. Different classes from the PDL are kept as separate as possible and are added one-by-one in Python with descriptive names. The model is built step-by-step and it is continuously checked whether it works well. A shuttle status log was used to print and track relevant state variables such as shuttle position, activity and current task at every simulation timestep. The logs enabled manual tracking of task executions, shuttle movements and operational phase transitions (e.g., Traveling → Positioning → Transfer). Next to that, the system received debugging print statements at essential checkpoints such as shuttle entry into the deadlock zone, task completion and many more. This allowed for the verification of control flow and state changes through manual step-by-step verification.

7.5.2. Deterministic Run

In simulation verification literature, deterministic testing is a standard method because it enables exact system behavior tracking through the removal of random elements [70]. The verification process involved setting all random number generators to use a specific seed value while replacing stochastic elements with fixed values for shuttle starting positions, half full racking and task generation. This approach, recommended by Fishman (2001) and Law & Kelton (2000), allows step-by-step tracing of simulation behavior [25, 48]. Six aspects were verified, including, among others, shuttle state transitions, task execution, control flow and deadlock recovery. Across ten deterministic runs these aspects remained fully consistent. These tests confirmed that the model behaves deterministically under identical inputs. A detailed breakdown of verified conditions and trace snapshots is provided in Appendix E.

7.5.3. Flow and Logic Consistency Checks

The verification of logical consistency between agents and system dynamics represents a basic requirement for discrete-event model verification. The following section describes four logic checks which include volume and flow consistency, zone/capacity constraint verification and verification of the deadlock handling and recovery. The checks verify that model behaviors follow operational rules while preventing simulation events from creating impossible or conflicting system states.

Volume and Flow Balance Checks

The verification process of simulation models requires two essential elements which include the conservation of entities inside the system and the verification of logical constraints for entity flow. Balci [4] identifies balance checks as essential DES methods to prevent unintended entity creation or loss during model runs. The developed SBSRS simulation models included internal counters that monitored the total number of items stored as $S(t)$ and retrieved as $R(t)$ and the items located in transit or temporary buffer as $W(t)$ at each simulation time step t. These variables are used to check the movements of the enities throughout the system. The model required verification of two main quantitative invariants to establish logical correctness:

1. Capacity Constraint: $R(t) \leq S(t)$
   The condition prevents retrieval operations from surpassing the stored item count.
2. Flow Balance: $S(t) - R(t) = W(t)$
   The condition confirms that the storage-retrieval difference equals the number of items which stay in the system but remain undelivered or uncompleted.

Both of the conditions were monitored continuously during the execution of a simulation to confirm if they are satisfied. Simulation running for 40 replications with 12000 time units with a 50/50 storage-retrieval task mix while maintaining both invariants throughout each time step:

- The capacity constraint was met during all time steps (100% compliance).
- The flow balance equation showed zero residual error because $|S(t) - R(t) - W(t)| = 0$ for all t.

This indicates that no tasks disappeared or were duplicated and that there was no miscounting.

Additional tests are conducted to check the behavior of the task execution. The trends of task execution were studied through regression analysis of cumulated data of completed tasks over time. Figure 7.3 shows the cumulative number of completed storage and retrieval tasks during the simulation period. The two plotted lines show the time-dependent growth of completed tasks with for storage (blue) and retrieval (orange) operations. The fairly parallel slopes of the two lines show that the tasks were distributed evenly and that the task processing rates remained constant. Linear regression lines were fitted to both data series to quantify this trend. Figure 7.3 presents the cumulative number of storage and retrieval tasks over time. The near-parallel and the linear increase of both of the lines indicates consistent throughput. Also, it shows that there are no big system bottlenecks such as deadlocks or unexecuted tasks. A linear regression analysis in Python generated the coefficient of determination $R^2_{\text{store}} = 0.996$ and $R^2_{\text{retrieve}} = 0.993$. This suggests that task execution happened at a stable and predictable rate throughout the simulation.

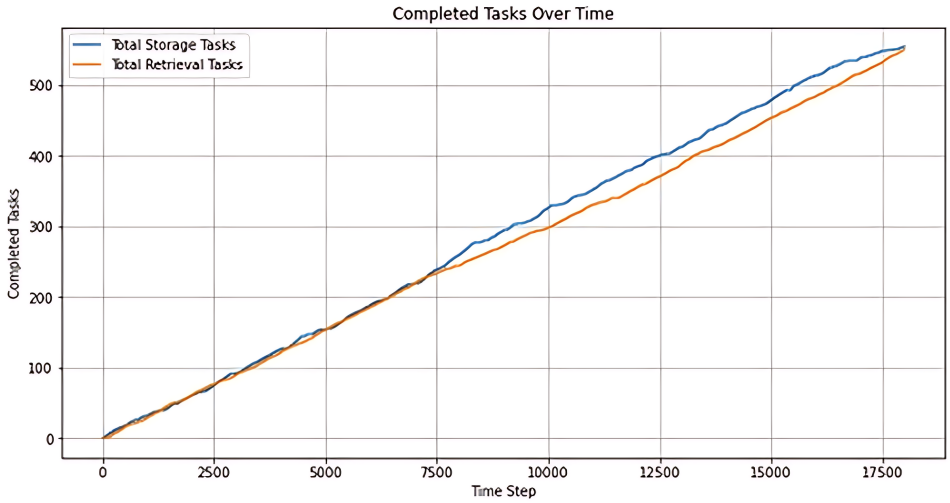


Figure 7.3: Cumulative Number of Storage and Retrieval Tasks over Time.

Next, the fairness among agents is checked. The evaluation of fairness in shuttle task assignment depends on the fairness ratio which can be calculated using the following standard formula:

$$\text{Fairness Ratio} = \frac{T_{\text{max}} - T_{\text{min}}}{T_{\text{mean}}}$$

where T_i is the number of tasks completed by shuttle i. The maximum number of tasks reached 54 while the minimum reached 50 and the average was 52 in the examined scenario. The fairness ratio of 7.7% falls below the 10% threshold which Law and Kelton [48] suggest for a balanced agent workload distribution. The visual representation confirms the calculated fairness ratio. Figure 7.4 presents the task completion distribution through a bar chart showing the number of tasks each shuttle finished. The small inter-agent variance confirms the consistency of the task dispatching mechanism.

Beside the logical flow and the fairness checks, broader system-wide balance conditions were verified through additional tests:

- The warehouse storage inventory remained within physical limits. The total storage items at any time t stayed within the available storage capacity range.
- The number of storage and retrieval tasks executed by shuttles are equivalent to the actual number of items handled which proved task-accounting integrity.
- The system started with a half-full rack that contained randomized items and the inventory behavior was as expected

$$\text{Storage}(t) = S_0 + S(t) - R(t)$$

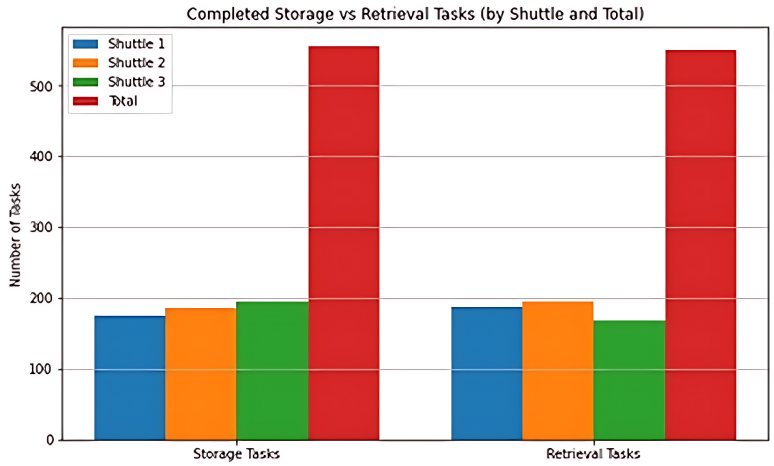


Figure 7.4: Distribution of Total Tasks executed per Shuttle.

where S_0 is the initial number of items in storage. The observed and computed values of Storage(t) matched at all time steps with zero deviation, indicating correct inventory state tracking.

All of the above combined confirm the key logic and flow tests. To summarize, the volume and flow balance checks produced results which validate the fundamental operational logic of the model. The model preserves items while keeping task rates steady, the model upholds physical capacity restrictions and achieves agent load distribution with minimal unbalanced conditions.

Spatial Constraints

The spatial constraints and routing logic throughout the simulation requires verification to maintain physical realism and avoid agent behavior that is infeasible. The verification of logic plays a vital role in SBSRS because zones need to be traversed in a constrained and exclusive manner. The detection of coordination failures including concurrent access or blocked zones can be achieved through zone-based tracking.

The simulation model 1 follows strict spatial constraints which include:

1. One-Shuttle-Per-Aisle Constraint: Not more then one shuttle per aisle is allowed.
2. Cross-Aisle Serialization: The cross-aisle functions as a common access point that requires simultaneous shuttle entry to be blocked. Again only 1 shuttle is allowed on the cross-aisle at any time
3. Transfer Zone Control: The Input and Output zones have their access serialized to prevent simultaneous task execution.
4. No Passing: Shuttles cannot pass eachother, even though they are never on the same node.

The simulation tracked shuttle zone occupancy at every time step to verify that all constraints remained intact. The occupancy trajectory extends across 300 time steps as shown in Figure 7.5. The colored lines in Figure 7.5 show which zone each shuttle visited at different times.

The following quantitative checks were performed across 10 simulation replications:

- Aisle Exclusivity: The system prevented any two shuttles from using the same aisle zone during the same time step.
  - *Total violations detected:* 0 / 3000 time steps = 0%
- Cross-Aisle Access Conflicts: The system prevented any shuttle from accessing or entering the cross-aisle at the same time.
  - *Max concurrent cross-aisle users:* 1 (expected), *violations:* 0
- Transfer Zone Overlap: The system allowed only one shuttle to access the Input or Output zone at a time. The CA-Safe zone required a single time-step delay before entry.

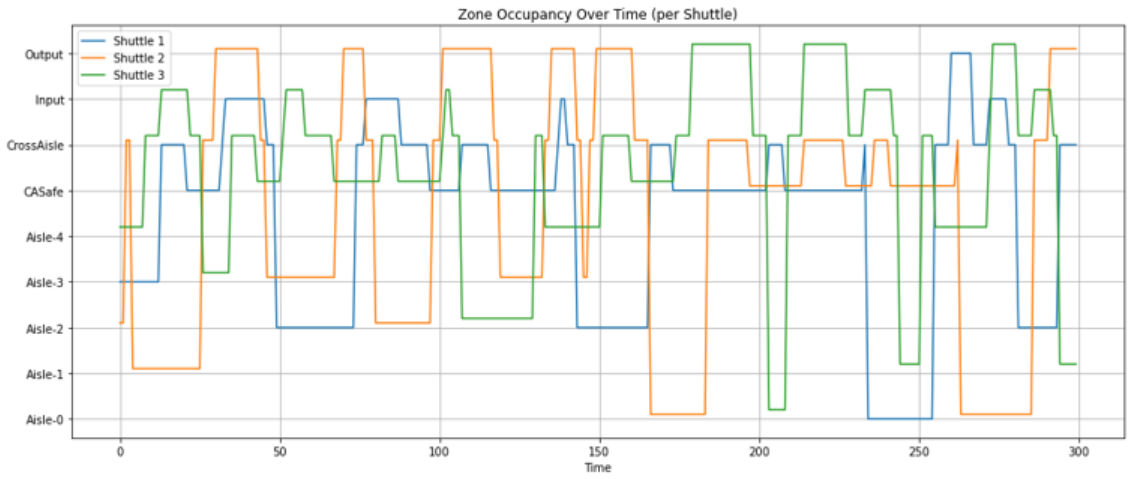


Figure 7.5: Zone Occupancy Over Time for Each Shuttle.

  - *Simultaneous occupancy detected:* 0 events over all runs

- No Passing: The system prevented two shuttles crossing each other (previous position shuttle one cannot be the same at current position shuttle two and vica versa)
  - *Swaps detected:* 0 events over all runs

In addition, the correctness of movement logic was verified. The storage network allows shuttles to move one node per time step and node-level reservations prevented any two agents from using the same node at the same time. The simulation follows the discrete-event node-based routing logic described in Banks et al. [5] and maintains both topological and temporal constraints at the most detailed level.

Next to that, the shuttle occupancy sequences demonstrated that shuttles left critical zones such as Cross-Aisle and Input/Output to enter CA-Safe or another idle zone which enabled space reuse and avoided deadlocks and congestion. The observed behavior demonstrates that the implemented zone control logic actively enforces mutual exclusion, priority and fairness rules. The graph of alternating occupancy in Figure 7.5 supports this conclusion. In the next section this will be discussed in more detail.

All of the above shows that the models meets the zone-based spatial constraints which ensures logical soundness in routing, access coordination and task execution. The observed zone behaviors show that no agent broke rules or traversed multiple nodes at once.

Deadlock Handling and Recovery Verification
A critical aspect of MAS verification is ensuring correct detection and resolution of deadlocks. In this model, deadlocks may occur when shuttles attempt to access shared zones simultaneously. To address this the simulation implements a deadlock handling protocol based on distance-based priority and predefined CA-Safe zones, as described in chapter 4.

The verification process confirmed that shuttles correctly detect deadlocks, yield or move to the safe zone when necessary and resume their tasks without loss of task data. The system consistently enforced the priority rule:

$$\text{Priority}(A) > \text{Priority}(B) \Leftrightarrow \text{Distance}(A) > \text{Distance}(B)$$

where $\text{Distance}(A)$ and $\text{Distance}(B)$ denote the number of hops between the current position of shuttle A and B and the nearest CA-Safe zone. This heuristic ensures that the agent farther from safety is granted right-of-way, while the closer agent yields and clears the path. Simulation traces

showed that this behavior occurred as expected in all test cases.

In all replications, CA-Safe zone logic, resumption accuracy and node locking were functioning as designed ensuring reliable recovery from spatial conflicts. The verification also ensured that agents do not get stuck in permanent blocking cycles [5]. The complete test setup and trace analysis used to verify deadlock handling are provided in Appendix E.

7.5.4. Sensitivity and Edge Case Testing

When analyzing the robustness and reliability of the simulation model a set of controlled sensitivity and edge-case scenarios should be tested. Such tests measure the response of the model to abnormal and extreme parameter values to verify if the model works as expected. Law and Kelton [48] state that testing for such conditions is necessary to build confidence in simulation-based systems' stability and resilience.

First, the model is tested for load storage versus retrieval task imbalance. Three different scenarios are created that examined storage against retrieval demand ratios at 80/20, 50/50 and 20/80. The warehouse operational extremes include fast storage filling and fast storage emptying, as well as constant or steady-state storage operations. In each scenario the model started with a half-filled rack, meaning 25 stored items. The minimum and maximum capacity of the storage rack are respectively 0 and 50.

The Figure 7.6 shown below illustrates how the item count changes over simulation time during each scenario. The three tests produce identical results: The storage bounds were not violated since the items in storage values stayed between 0 and 50 ($0 \leq$ Items in Storage ≤ 50). The task generators are following the inventory restrictions since all of the retrieval requests remain below the available items in storage. The item flow transitions were smooth and monotonic without any oscillations or abnormalities in logic.

- Storage-dominant scenario (80/20): The system expands its storage capacity until reaching 50 items before stopping at maximum capacity. The storage curve becomes flat because storage locations reach their maximum capacity which shows the system functions correctly.
- Balanced scenario (50/50): The storage item quantity operates within normal operational limits in the 50/50 scenario by staying between 25 and 35. The system operates within established boundaries to avoid both item accumulation and supply shortages.
- Retrieval-dominant scenario (20/80): The amount of items in storage quickly decreases until reaching zero. The model is following inventory rules by preventing to execute any retrieval tasks that exceed existing stock quantities.

The results show that the task handling and storage control logic work reliably under different demand intensities.

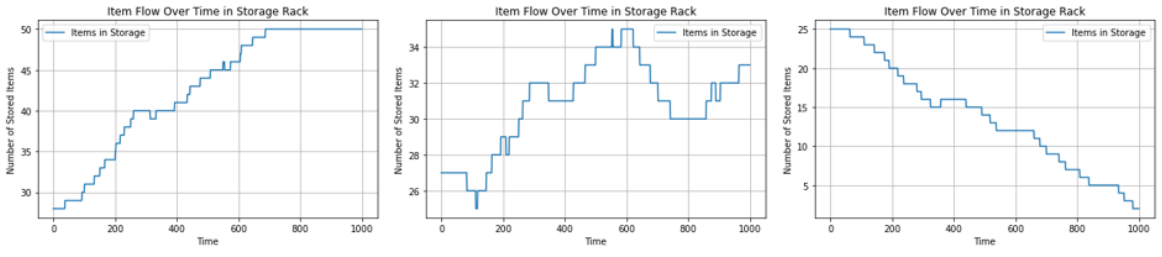


Figure 7.6: Item Flow Over Time for Different Storage/Retrieval Ratios: 80/20 (left); 50/50 (center); 20/80 (right).

Besides the load stress testing some additional degeneracy tests are conducted. The following scenarios were tested, as suggested by Sargent, to confirm its behavior during extreme or boundary conditions [70]:

- Zero tasks: The simulation was initialized with zero storage tasks and zero retrieval tasks. The

shuttles remained idle at their starting positions without any movement which confirms that there was no unintended agent behavior or background tasks in the simulation.
- Empty rack startup: The system was initialized with no items in storage and high retrieval demand. All retrieval tasks were deferred and no shuttle attempted a retrieval without valid input stock. Storage tasks were processed normally as items arrived.
- Simultaneous cross-aisle access: Three shuttles were routed to converge on the cross-aisle simultaneously. Serialized access was correctly enforced and no overlapping nodes or collisions occurred.
- Zero shuttles: The simulation was initialized without any active shuttle agents. All tasks remained in queue. The simulation did not crash and correctly handled the absence of shuttles.

In all cases, the model behaved in accordance with the conceptual design expectations. No logic errors, deadlocks, or invalid shuttle states were registered. These results demonstrate that the simulation model is not only functionally correct under nominal conditions but also resilient to edge cases and extreme input configurations.

7.6. Experimental Plan

The verified simulation model from section 7.5 serves as the basis for this chapter to establish the experimental plan which evaluates the performance of the hierarchical and hybrid control architectures in the SBSRS environment. The goal is to develop a scientifically valid and statistically sound experimental design which evaluates the architectures under different operational conditions.

The experimental design defines simulation parameters including warm-up period, run length and number of replications to achieve statistical robustness of the results. The research follows established guidelines from simulation output analysis literature [71, 48, 68]. The experimental plan consists of controlled experiments to analyze system performance by adjusting parameters like system scale and shuttle density. Doing so, it can be assessed how well each architecture handles system complexity and rising agent interactions since these represent essential challenges from chapter 2. The framework applies identical scenarios for both architectures to establish performance differences that stem from the control structure.

7.6.1. Simulation Setup

The following sections describes the fundamental simulation parameters which apply to all experiments. The experimental design includes a determined warm-up period, run length and the required number of replications to ensure statistical validity of simulation outputs. The following subsections explain these parameters through established methods from simulation output analysis literature.

Initialization

Before the simulation starts, the SBSRS environment receives its initial setup through warehouse layout definition, shuttle agent initialization and system resource configuration. The simulation environment operates within a fixed time horizon using SimPy as its programming framework. The simulation requires the following essential initialization procedures:

- System network construction: The SBSRS contains a directed graph structure at one level which uses nodes to represent storage slots and cross-aisles and input/output points and edges to show shuttle movement paths. The network creation happens automatically through user-defined system parameters.
- Shuttle generation: The simulation creates agents to represent shuttles which receive an ID, a random starting positions and task execution capabilities. The decision-making and logic for shuttles are described in the simulation control architecture earlier in this chapter and chapter 4.
- Task generation: The simulation initiates a task generator which produces storage and retrieval requests during the simulation period. The control logic operating during each experiment determines which tasks will be assigned to shuttles.
- Item generation: The simulation starts with a rack with 50% of its capacity filled by randomly positioned items.

The initialization framework allows simulation runs to start with a uniform system state but keeps variability through stochastic elements like task generation, initial item distribution within the rack and initial shuttle positions.

Warm-Up Period
To ensure that performance measurements are not biased by initialization effects, this study applies a formal warm-up analysis using Welch's graphical method [83]. The method follows Law's recommendation to detect steady-state onset empirically because it removes transient bias from performance estimation [47]. The Welch's method revealed that a 1200 simulation second warm-up period should be used to eliminate initialization bias [83]. This graphical method evaluates the stabilization of a key performance indicator, in this case system throughput, over time.

The throughput trend, along with its 95% confidence interval (CI), is shown in Figure 7.7. The system shows large mean variations together with wide CI bands during its initial transient phase. The throughput curve reaches a flat state while the CI bands become narrower at the 1200-second mark which indicates steady-state behavior has been reached. The vertical red dashed line in the figure indicates the specific point in time at which the confidence interval first exceeded 95%.

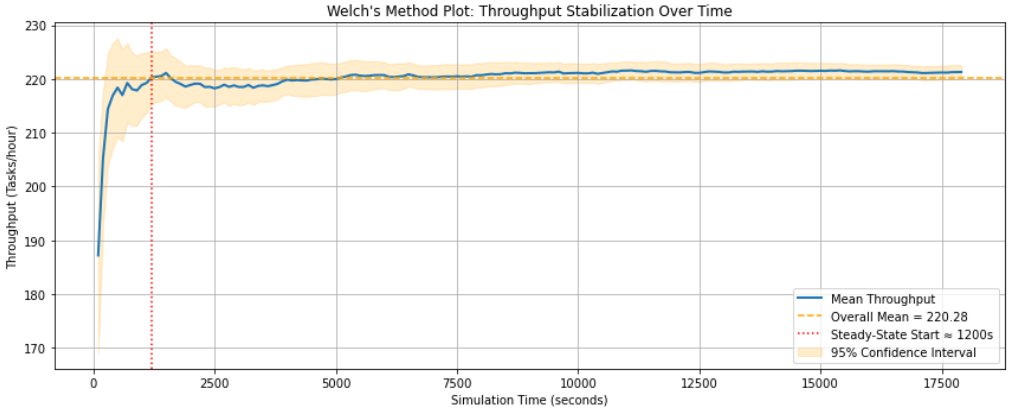


Figure 7.7: Welch's Method Plot: Throughput Stabilization Over Time

Run Length
After establishing the warm-up period the total run length was determined according to Law [47] and Rossetti's [68] common rule of thumb which requires the simulation to run for at least 10 times the warm-up duration to achieve steady-state observation [5].

$$\text{Run Length} = 10 \times 1200 = 12000 \text{ time units} \tag{7.1}$$

Number of Replications
Tho ensures that the estimated performance metrics such as throughput are statistically significant and sufficiently precise, a simulation experiments are replicated a number of times. The analysis of stochastic variability required 40 independent replications to be performed. The required number of replications was determined through a replication convergence analysis.

Figure 7.8 shows the cumulative mean throughput together with its 95% confidence interval as a function of replication count. The analysis followed the convergence criterion proposed by Law [27], which specifies that the relative CI width should fall below 5%:

$$\text{Relative CI width} = \frac{2 \cdot \text{CI}_{\text{half-width}}}{\bar{x}} \cdot 100\% \tag{7.2}$$

Where $\bar{x}$ is the sample mean throughput and $\text{CI}_{\text{half-width}}$ is computed using the fixed-width CI formula:

$$\text{CI}_{\text{half-width}} = t_{\alpha/2,\, n-1} \cdot \frac{s}{\sqrt{n}} \tag{7.3}$$

Where s is the sample standard deviation of throughput across replications; n is the number of replications; and $t_{\alpha/2,n-1}$ is the critical value from the Student's t-distribution for a 95% confidence level.

The amount of replications should continue until the CI relative width reached below 5%, which meets the convergence requirement specified in [47]. As can be seen in Figure 7.8, the relative CI width reaches 5% relative width at 20 replications, satisfying the convergence criterion proposed by Law [47]. The value was doubled following to the rule-of-thumb [47]:

$$n_{\text{final}} = 2 \times n_{\text{CI}\leq 5\%} = 2 \times 20 = 40 \text{replications} \tag{7.4}$$

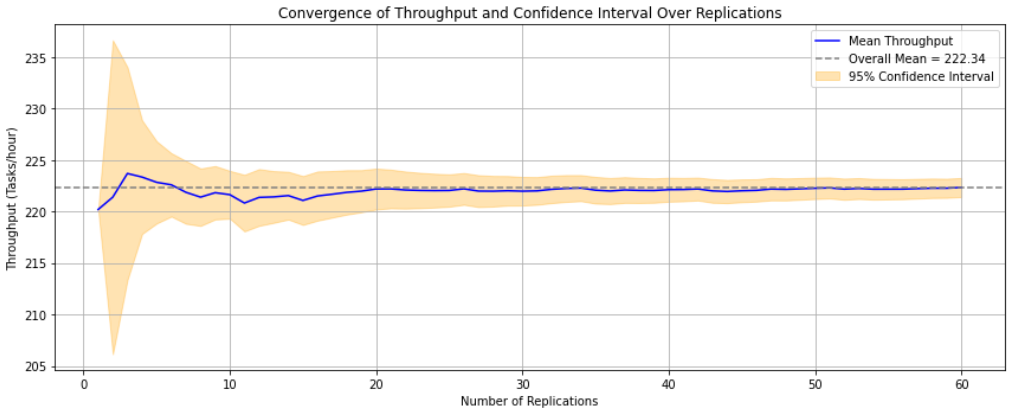


Figure 7.8: Replication Convergence Plot for Hierarchical Model (Throughput Mean and 95% CI over Replications).

All in all, the final simulation setup by following best practices in simulation modeling and align with recommendations from literature, is the following:

- Warm-up period: 1200 seconds (Welch's method)
- Run length: 12000 seconds (rule of thumb 10 × warm-up)
- Number of replications: 40 (based on CI convergence and doubling rule of thumb)

7.6.2. Baseline Test Scenario and Configuration

The experimental scenarios are evaluated against a baseline system configuration which represents normal operational conditions of a single-level SBSRS. The baseline setup functions as a reference point to maintain uniform performance evaluation between control architectures and experimental conditions.

The baseline simulation scenario includes a fixed warehouse layout, standard task load, and moderate system resources. The initial rack is filled up to 50% with randomized items at randomized locations to model semi-operational status. The shuttles start from randomized positions to create system variability. Table 7.2 shows an overview of the default SBSRS configuration parameters. The experiments begin with this configuration unless a specific scenario dictates otherwise.

All shuttle agents maintain identical behavioral parameters to achieve consistent behavior across control architectures, as shown in Table 7.3. They maintain a fixed speed of 1 node per time unit throughout all scenarios. The discrete nature of the simulation environment together with this modeling choice enables event scheduling while maintaining behavioral fidelity [12]. The unit-based movement model provides uniform network traversal costs to all agents which enables reproducible architectural comparisons. Agent-based and DES models of shuttle-based systems commonly use fixed-speed movement as a standard simplification when evaluating coordination logic instead of detailed kinematics [33]. The parameters demonstrate actual shuttle behavior while keeping the comparability under control. The model does not include acceleration or deceleration functions because movement occurs through discrete unit steps.

Table 7.2: Default SBSRS Configuration Parameters

Parameter	Value
Number of aisles	5
Slots per aisle	10
Input nodes	1
Output nodes	1
Cross-aisle configuration	Shared, single-level
Initial rack state	Half-full, randomly distributed
Task generation	50/50 (storage vs retrieval tasks)
Shuttle agents	2 shuttles
Shuttle start positions	Randomized
Control assignment	Varies by architecture
Simulation time	12000 time units
Warm-up period	1200 time units
Number of replications	40 per scenario

Table 7.3: Shuttle Behavioral Parameters

Parameter	Value
Movement speed	1 node per time unit
Cross-aisle entry delay	1 time unit
Positioning time	2 time units
Item transfer time	2 time units (for both storage and retrieval)
Acceleration modeling	Not included (uniform step movement)

7.6.3. Scenarios for Comparative Experiments

To compare the performance of hierarchical and hybrid control architectures, a structured set of scenarios was designed to systematically vary system configurations and system complexity. All scenarios use the same baseline configuration and parameters introduced in previous sections, ensuring that differences in performance stem solely from architectural design. To assess the impact of control architecture on SBSRS performance, the experimental scenarios are defined along two main dimensions: system size and shuttle density.

System size is scaled by expanding both the number of aisles and storage slots per aisle, resulting in four warehouse configurations: 10×10, 15×15, 20×20 and 25×25. This allowed the analysis to explore how each architecture responds to increasing routing complexity and spatial scale, particularly in terms of throughput degradation, coordination overhead and congestion distribution.

Shuttle density was adjusted by simulating between two and six concurrently operating shuttles per system. This tested how well each architecture handles increased agent interactions, especially in shared resources such as the cross-aisle. As the number of shuttles grows, the pressure on decision-making logic, routing stability and resource contention management increases. These are key stressors identified in chapter 2 as limitations of conventional SBSRS control.

Combined with the standardized warm-up period, run length and number of replications defined earlier in this chapter, the design enables robust, statistically valid comparisons between the hierarchical and hybrid models. Each architecture was tested across all 20 scenario configurations. The experimental matrix below provides an overview of the system sizes and shuttle counts tested:

Table 7.4: Experimental Matrix Overview

System Size (Aisles × Slots)	Shuttle Counts Tested
10 × 10	2, 3, 4, 5, 6
15 × 15	2, 3, 4, 5, 6
20 × 20	2, 3, 4, 5, 6
25 × 25	2, 3, 4, 5, 6

In addition to the fixed matrix of scenarios described above, an additional experiment was included to determine the maximum achievable throughput per system size for each control architecture. While the main scenarios test both models under identical shuttle counts, this supplementary experiment identifies the best-performing shuttle configuration per warehouse size. It allows each architecture to operate at its optimal point thereby revealing its full performance potential. This comparison supports a fairer assessment of architectural scalability and control efficiency under best-case conditions. The results of this experiment are presented and discussed in chapter 8.

7.6.4. Key Performance Indicators

The goal of this study is to evaluate the impact of control architecture on the operational performance of SBSRS. In the context of SBSRS operating under I4.0 principles, operational performance refers to the system's ability to maintain high task throughput, resource efficiency and behavioral stability as system conditions grow more complex. Or in other words, operational performance in SBSRS encompasses the system's ability to efficiently, reliably and predictably execute storage and retrieval tasks under varying system scales and resource configurations. To objectively assess and compare the two control architectures, this research focuses on four key performance indicators (KPIs): throughput, scalability, efficiency and robustness. These indicators have been consistently used in SBSRS and automated warehouse research to evaluate control system behavior under dynamic conditions [33, 52, 9]. The selection of KPIs is grounded in the scientific literature and tailored to capture the multi-faceted nature of operational performance, as well as the specific benefits I4.0-oriented architectures are expected to deliver.

System throughput is the primary KPI, defined as the number of completed storage and retrieval tasks per hour of simulated time. One simulation time unit corresponds to one second. Throughput is calculated as follows:

$$\text{Throughput (tasks/hour)} = \frac{N_{\text{tasks}}}{T_{\text{sim}}/3600} \tag{7.5}$$

where N_{tasks} is the total number of completed tasks and T_{sim} is the total simulation time in seconds. Algorithm 4 in Appendix D, shows the pseudococde of how this KPI is calculated in the simulation. Throughput is the most fundamental output measure for automated storage systems, as it directly captures the system's capacity to deliver value in terms of processing tasks under given physical and control constraints [33, 52].

Efficiency accounts for layout-related effects by normalizing throughput with shuttle travel effort. Recognizing that physical layout and shuttle travel distance can bias throughput comparisons, this research also incorporates efficiency metrics based on normalized throughput. It is defined as:

$$\text{Efficiency (tasks/m)} = \frac{N_{\text{tasks}}}{\bar{d}_{\text{task}}} \tag{7.6}$$

where $\bar{d}_{\text{task}}$ is the average distance traveled by shuttles per completed task. This normalization allows for fairer benchmarking between systems of different sizes or layouts and directly assesses how effectively each control strategy translates shuttle movement into productive work [52]. Additionally, throughput per shuttle is considered, reflecting the marginal utility of additional shuttles and identifying potential bottlenecks or diminishing returns as resource levels increase.

Scalability is assessed by observing how throughput and efficiency respond to increasing system size (aisles and slots) and shuttle density. A scalable architecture maintains or improves performance as operational complexity increases. Throughput alone does not provide the full picture, especially when comparing systems of different sizes or resource levels. Therefore, scalability is explicitly analyzed by examining how throughput and efficiency change as the number of aisles, slots or shuttles increases. Scalability is crucial for I4.0-oriented systems which are expected to accommodate future growth and dynamic adaptation [8]. To operationalize scalability, throughput is measured across multiple system configurations, revealing whether the architecture supports performance retention or improvement at larger scales or higher shuttle densities.

Robustness measures the consistency of system performance under stochastic variation. It is quantified by the coefficient of variation (CV) of throughput across independent replications:

$$\mathrm{CV} = \frac{\sigma_{\mathrm{tp}}}{\mu_{\mathrm{tp}}} \tag{7.7}$$

where μ_{tp} and σ_{tp} are the mean and standard deviation of throughput, respectively. Lower variance indicates that the architecture delivers consistent results and is less sensitive to random events, disturbances or localized congestion, a key attribute for operational resilience in automated warehouses [63]. Throughput variance is visualized with box-plots and reported alongside mean values to provide a comprehensive assessment of system predictability.

These KPIs enable a scientifically grounded comparison between hierarchical and hybrid architectures. Their integration ensures that the evaluation captures not only raw output but also the architectures' ability to scale, utilize resources efficiently and maintain performance stability under increasingly complex conditions.

7.7. Conclusion

This chapter addressed sub-research question 5: *How can the hierarchical and hybrid control architecture for SBSRS be implemented and verified in a simulation model?* It presented the development, structure and verification of the simulation model used to evaluate the two control architectures studied in this research: the current hierarchical architecture and the proposed hybrid MAS-based alternative. Using a hybrid DES and ABM approach implemented in Python the model captures both centralized and distributed control dynamics within a single-level SBSRS layout.

The model scope was carefully defined to isolate shuttle-level behavior under controlled, steady-state conditions enabling a fair architectural comparison. All model components, including the network-based layout, shuttle agents, task generation and shared resources, were implemented consistently across both architectures with architectural logic integrated as modular extensions. Architecture-specific behaviors such as task allocation, path planning and conflict resolution were designed to reflect the structural differences between centralized and agent-based decision-making.

Extensive verification procedures confirmed the correctness, consistency and robustness of the model under deterministic, stochastic and edge-case conditions. The simulation framework was then used to design a statistically grounded experimental plan based on warm-up analysis, replication convergence and scenario variation along key complexity axes such as system scale and shuttle density.

Altogether, this chapter established a verified, modular simulation that allows for an objective and repeatable comparison between the two control architectures. The subsequent chapter presents and analyzes the simulation results offering insights into how architectural design influences throughput, scalability, efficiency and robustness in SBSRS environments.

8

Simulation Results & Discussion

This chapter presents the results of the simulation experiments conducted according to the experimental plan outlined in section 7.6. The goal is to quantitatively evaluate and compare how the hierarchical and hybrid architectures perform under different system conditions. The plan tested the hierarchical and hybrid control architectures under systematically varied operational conditions, combining four system sizes (10×10, 15×15, 20×20, 25×25) with shuttle fleets ranging from two to six vehicles. Each scenario was simulated for 12,000 seconds with a 1,200-second warm-up period and 40 replications ensuring statistical validity. The evaluation is structured around four key performance indicators (KPIs) defined in Chapter 7: throughput (capacity), normalized efficiency (throughput per distance unit), scalability (sensitivity to system size and density) and robustness (performance variance). Doing so, the chapter addresses the following sub-research question:

Sub-RQ 6: What is the impact of the hybrid control architecture on the operational performance of a SBSRS compared to the hierarchical control architecture?

The following sections present the results in line with these KPIs. To avoid redundancy detailed descriptive analyses of the architectures in isolation (see Appendix F) are not repeated here but are used as supporting evidence. Instead this chapter focuses on the comparative evaluation. In addition to the quantitative results a discussion section interprets the observed performance patterns in relation to system design principles, literature expectations and the improvement needs defined earlier in the study. Finally, the limitations of the simulation are reviewed to clarify the scope and applicability of the findings.

8.1. Comparative Throughput Performance

System throughput is the main measure of SBSRS capacity under different operating conditions. Figure 8.1 shows how hybrid control improves throughput compared to hierarchical control, reported both as percentages and as absolute values in tasks per hour. Across all scenarios hybrid control outperforms hierarchical control. In a small 10×10 system with two shuttles it completes about 24 more tasks per hour (+15%). In larger and denser systems this advantage becomes much stronger reaching more than 52 extra tasks per hour (+90%) in a 25×25 system with six shuttles.

The results show two clear trends. First, relative performance gains grow with system size because centralized coordination in hierarchical control increasingly becomes a bottleneck. Second, the difference grows even more as shuttle density rises: while hierarchical control struggles with congestion and queuing, hybrid control uses distributed decision-making to keep shuttles moving in parallel more effectively.

Overall, these findings show that hybrid architectures do not only reach higher throughput but also handle growth in system size and complexity more effectively. By enabling local decision-making and reducing coordination overhead, hybrid control avoids much of the congestion and waiting that limit hierarchical systems [8, 53].

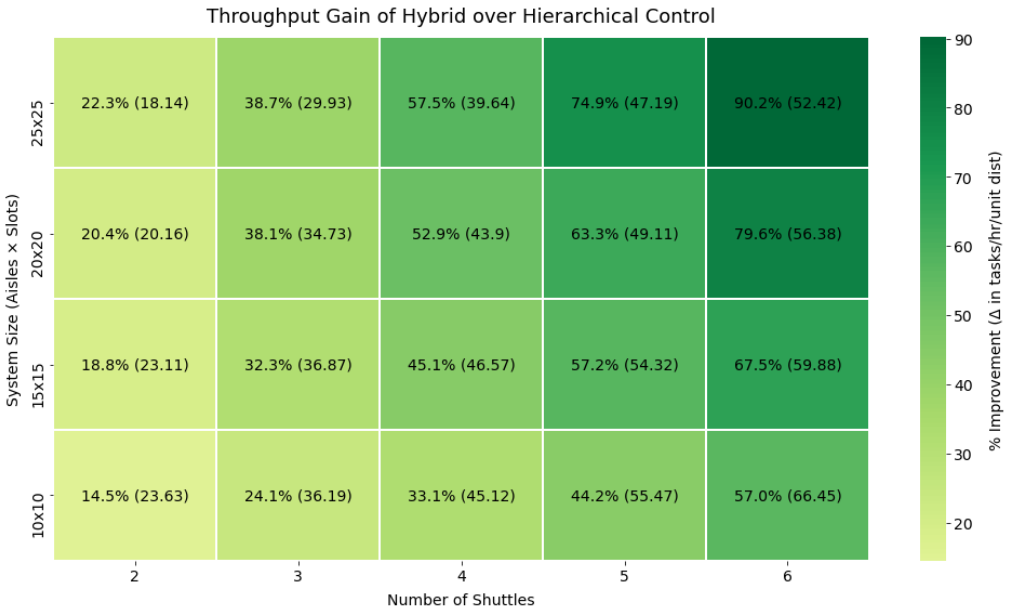


Figure 8.1: Heatmap of throughput gain achieved by hybrid control compared to hierarchical control across different system sizes and shuttle counts. Relative efficiency improvements grow with system scale and shuttle density.

8.2. Scalability

The heatmap in the previous section already showed that hybrid architectures achieve consistently higher throughput than hierarchical control, especially as system size and shuttle density increase. To better understand why this happens, throughput scalability is examined in more detail along two dimensions: adding more shuttles in a fixed layout and increasing warehouse size with a fixed number of shuttles.

Scalability across shuttle density

Figure 8.2 shows how throughput changes as more shuttles are added in a 20×20 warehouse. For hierarchical control throughput peaks at 98 tasks/hour with two shuttles but then steadily declines, dropping to 70 tasks/hour at six shuttles. This decline is caused by congestion and path conflicts that the centralized controller cannot resolve effectively at higher densities.

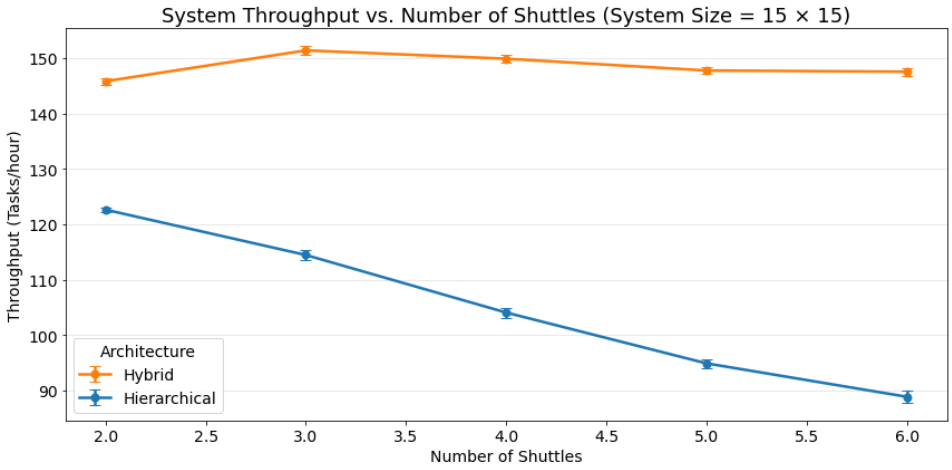


Figure 8.2: System throughput as a function of shuttle count in a 20×20 layout. Hybrid control continues to scale with additional shuttles, reaching 128 tasks/hour at five shuttles, while hierarchical control peaks at 98 tasks/hour with two shuttles and declines as shuttle density increases.

Hybrid control performs very differently. Throughput increases with additional shuttles reaching 128 tasks/hour with five shuttles and then flattens slightly to 126 tasks/hour at six shuttles. This stable performance under dense conditions highlights the strength of distributed routing and local negotiation which allow shuttles to coordinate more effectively and keep tasks running in parallel. The performance gap between the two architectures also grows with density: from about +20% at two shuttles to almost +80% at six shuttles (a gain of 56 tasks/hour). Importantly, this difference comes less from rapid gains in hybrid performance and more from the decline of hierarchical control as congestion builds up.

Scalability across system size
Figure 8.3 shows how throughput changes as system size increases with the fleet size fixed at three shuttles. For both control architectures throughput steadily decreases as the warehouse grows since shuttles must travel longer distances to complete each task.

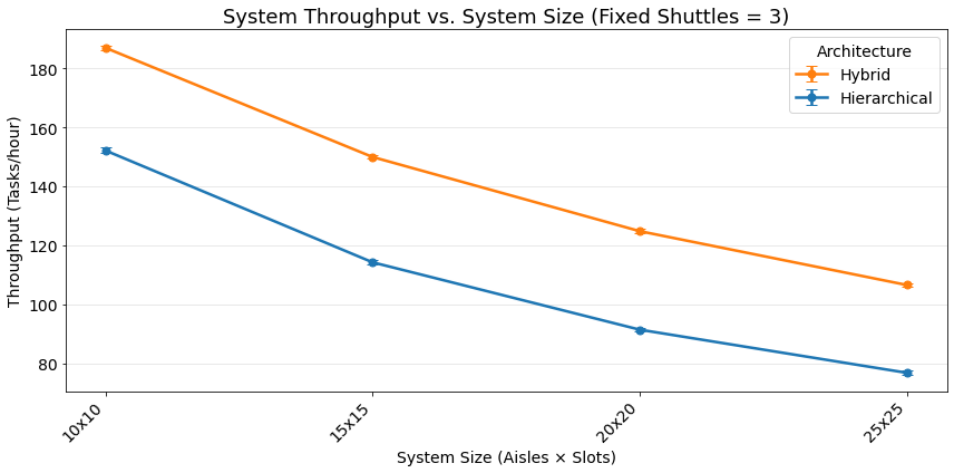


Figure 8.3: System throughput as a function of system size with three shuttles. Throughput decreases for both architectures as warehouse size grows due to longer travel distances, but hybrid consistently maintains higher throughput across all system sizes.

Under hybrid control throughput falls from 187 tasks/hour in the 10×10 layout to 107 tasks/hour in the 25×25 system. Hierarchical control follows the same pattern dropping from 152 to 76 tasks/hour. In absolute terms both lose almost the same amount of throughput (around 76-80 tasks/hour). However, the decline is slightly more gradual for the hybrid model suggesting that it handles the increase in travel distances a little more efficiently.

The key finding is not that hybrid declines much slower but that it consistently maintains higher throughput across all system sizes. Even as larger layouts reduce overall performance, the hybrid model preserves a clear advantage showing that it can make better use of available resources as systems grow.

Together, these scalability results explain the widening performance gap seen in the throughput heatmap. Hierarchical control cannot scale with higher shuttle densities and quickly loses performance in larger systems, while hybrid control sustains high throughput by reducing congestion and coordinating resources more effectively. This scalability advantage shows why hybrid architectures are more suitable for large high-density SBSRS environments.

8.3. Operational Efficiency

While raw throughput measures overall task completion, it does not account for the travel effort required to achieve it. To enable a fair comparison of operational performance across system sizes and shuttle configurations normalized throughput is used, defined as tasks per hour per unit shuttle travel distance. This metric reveals how effectively each control architecture converts movement into productive work.

Figure 8.4 shows the efficiency improvement of hybrid over hierarchical control, expressed as normalized throughput across system sizes and shuttle counts. The heatmap reports both relative (%) and absolute (tasks/hour per distance unit) gains making it possible to compare architectures fairly across different scales. The results show that hybrid control consistently delivers higher normalized throughput across all tested system sizes and shuttle counts with the advantage becoming particularly clear in large and dense systems. In compact layouts such as a 10×10 system with two shuttles hybrid control improves movement efficiency by about 16% compared to hierarchical control. As system size and shuttle density increase, this relative advantage grows steadily reaching up to 68% in a 25×25 layout with six shuttles.

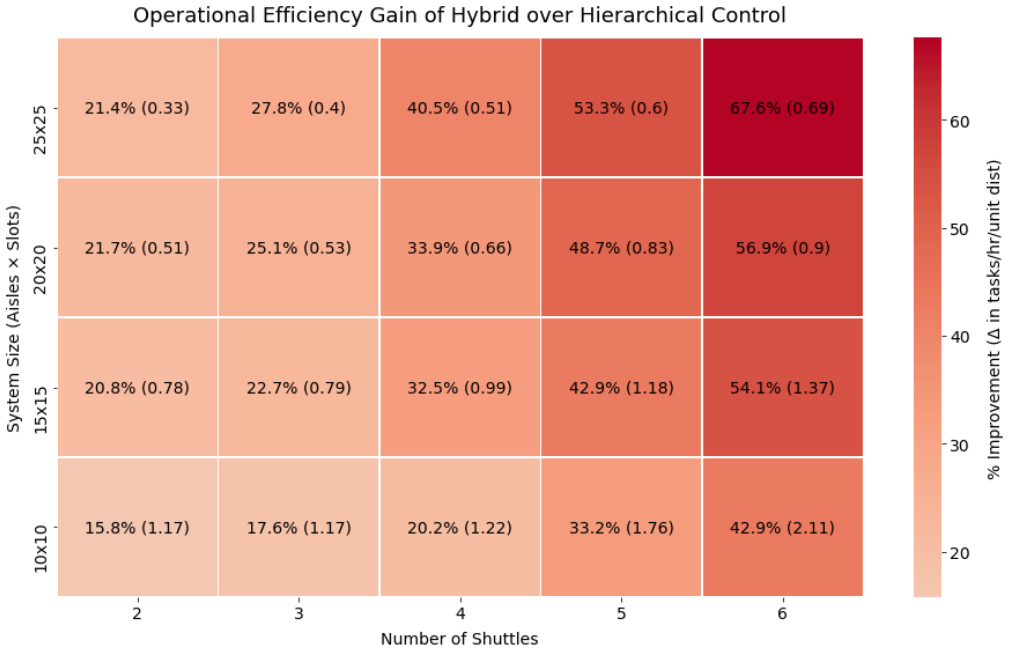


Figure 8.4: Heatmap of normalized throughput gain achieved by hybrid control compared to hierarchical control across different system sizes and shuttle counts. Relative efficiency improvements grow with system scale and shuttle density.

Three main trends can be seen in the analysis. First, efficiency gains increase with shuttle count, as hybrid coordination prevents the traffic jams and blocking behavior that occur under centralized logic. Second, efficiency gains also grow with system size: although efficiency inevitably drops in absolute terms as shuttles must travel longer distances per task, hybrid control reduces this wasted effort much more effectively than hierarchical control. Finally, absolute efficiency still decreases with system size because of longer travel distances but the hybrid model manages this decline much better preserving higher levels of productivity.

Overall, hybrid MAS does not just increase output, it dramatically improves how effectively shuttles use their time and movement especially in large dynamic systems.

8.4. Best Achievable Throughput and Efficiency

The heatmaps in Figures 8.1 and 8.4 compare both control strategies under the same number of shuttles and the same system size. However, this does not show the fact that each architecture may perform best at a different fleet size. To address this, Figures 8.5 and 8.6 present the best achievable throughput and normalized throughput for each system size. This is based on the optimal shuttle count of both models. This approach provides a benchmark of the scalability ceiling of each architecture and shows how well they can make use of extra resources.

The results show that hybrid control consistently achieves higher maximum throughput in all warehouse sizes. The relative improvement increases with scale: +14% in 10×10 systems, +23% in 15×15,

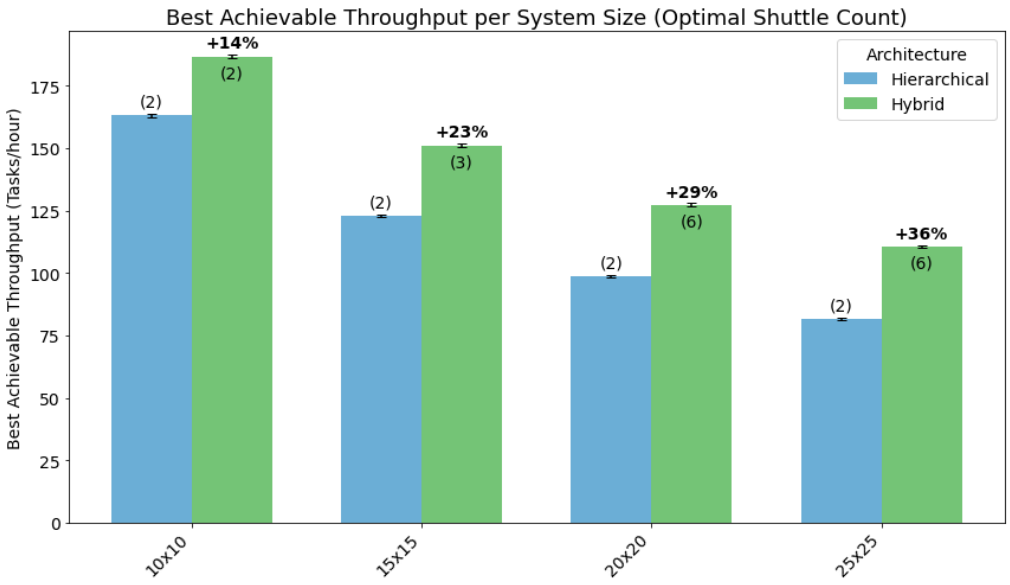


Figure 8.5: Best achievable throughput per system size, determined at the optimal shuttle count for each architecture (shown in parentheses above the bars). Hybrid control consistently outperforms hierarchical control, with gains increasing from 14% in small systems to 36% in large ones.

+29% in 20×20, and +36% in 25×25 layouts. In many cases hybrid reaches its best performance at higher shuttle counts than hierarchical control showing its ability to handle larger fleets without performance saturation.

Efficiency results, Figure 8.6, show a similar trend. Best normalized throughput improves by +16% in 10×10 systems, +21% in 15×15, +22% in 20×20, and +21% in 25×25. This indicates that hybrid control not only raises raw throughput but also ensures that additional shuttle activity leads to more productive work rather than wasted travel.

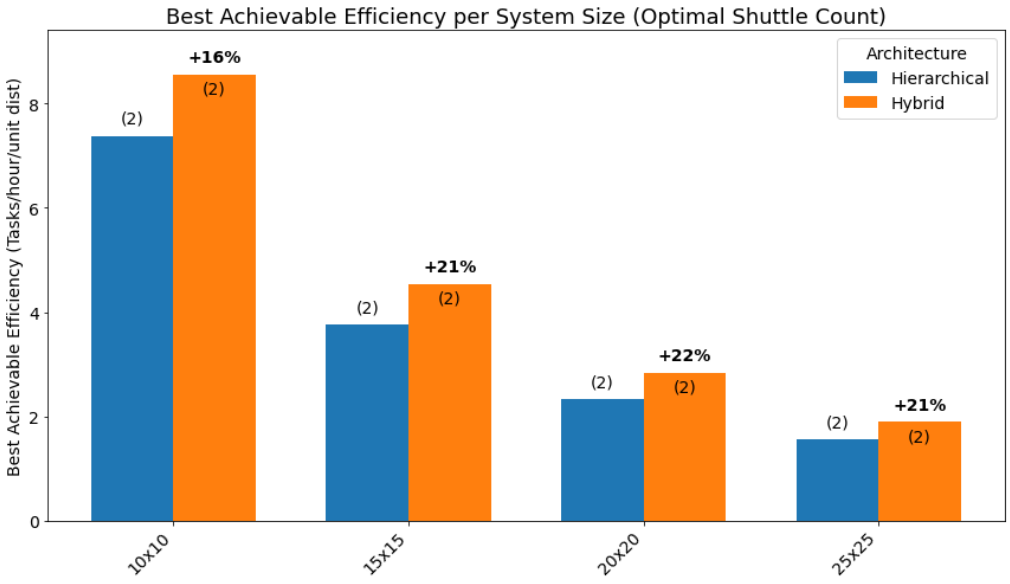


Figure 8.6: Best achievable efficiency per system size, determined at the optimal shuttle count for each architecture (shown in parentheses above the bars). Hybrid control maintains a consistent advantage of 16-22% across all tested system sizes.

Overall, these findings demonstrate that hybrid MAS architectures scale more effectively and main-

tain stronger performance even when both systems are compared at their optimal operating points. This supports the conclusion that partial distributed decision-making improves both capacity and efficiency in SBSRS environments and offers a more reliable solution for large-scale and high-density systems.

8.5. Robustness

Robustness refers to how stable and predictable system performance remains when the same conditions are repeated. In industrial settings this is highly important because warehouses rely on consistent throughput and low variability to plan operations and ensure reliability. In this study robustness was evaluated by looking at throughput distributions across 40 independent replications per scenario. Two measures were used: the interquartile range (IQR) which shows the spread of the central 50% of results and the coefficient of variation (CV) which measures variability relative to mean performance.

Figure 8.7 shows the throughput distributions for a representative 15×15 system across shuttle densities from two to six. The results show that both architectures maintain relatively narrow IQRs, typically between 6 to 8 tasks/hour across all shuttle counts. For example, with four shuttles the IQR was 6.8 tasks/hour for the hybrid architecture and 7.4 tasks/hour for the hierarchical one; with six shuttles these values were 7.1 and 8.0 respectively. At first sight these differences may seem minor but the boxplots reveal that the hybrid consistently produces tighter interquartile ranges, shorter whiskers and fewer outliers. This means that not only the middle 50% of results are closer together but also that extreme cases deviate less which leads to more predictable outcomes. By contrast, the hierarchical model shows both high and low throughput outliers reflecting its greater sensitivity to stochastic variation. Low outliers are typically caused by congestion or near-deadlock events while high outliers represent unusually favorable task sequences.

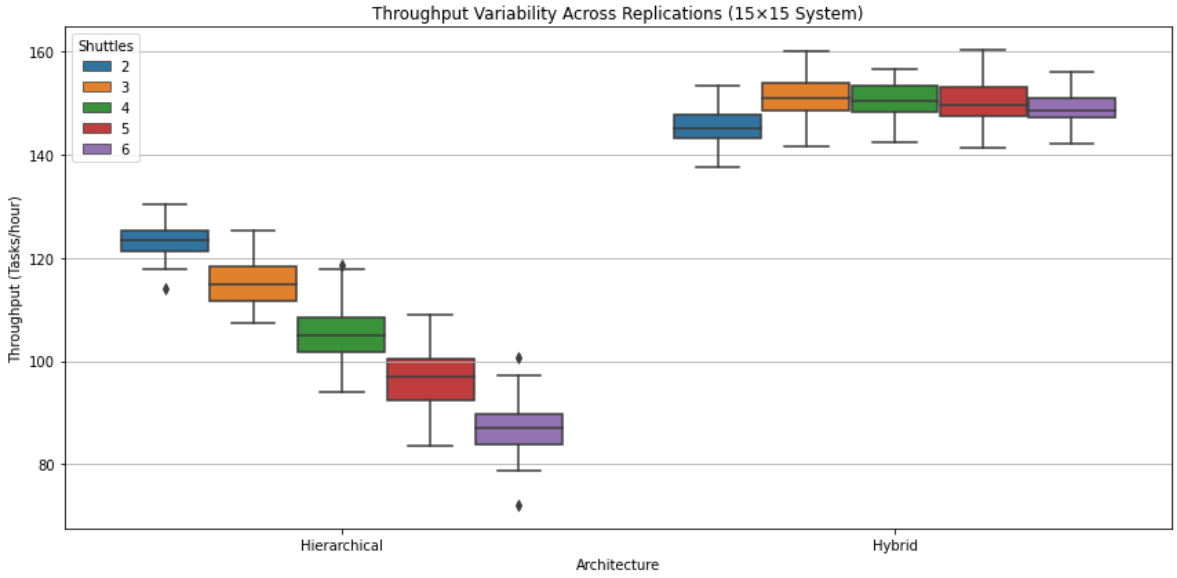


Figure 8.7: Boxplot of throughput distributions for hierarchical and hybrid architectures at different shuttle counts in a 15×15 system. Hybrid shows tighter interquartile ranges, shorter whiskers, and no outliers, indicating higher robustness compared to hierarchical control.

The CV analysis highlights the differences more clearly (Table 8.1). For the hybrid architecture, CV values remained consistently low between 2.2-2.8% across all shuttle counts. In comparison, the hierarchical model started at 2.7% (2 shuttles) and increased steadily to 6.0% (6 shuttles). Although these percentage differences might appear small literature on simulation output analysis emphasizes that even shifts of one or two percentage points in CV can represent substantial differences in robustness, especially in tightly coupled systems where variability compounds over time. Using CV instead of standard deviation also ensures a fairer comparison as it accounts for different absolute throughput

levels.

Table 8.1: Throughput statistics (mean, standard deviation, and coefficient of variation) for hierarchical and hybrid architectures in a 15×15 system over 40 replications.

Shuttles	Hierarchical			Hybrid		
	Mean [tasks/h]	Std [tasks/h]	CV [%]	Mean [tasks/h]	Std [tasks/h]	CV [%]
2	123.23	3.38	2.74	145.59	3.30	2.27
3	115.12	4.76	4.14	151.18	3.99	2.64
4	104.83	5.91	5.64	150.18	3.91	2.60
5	96.52	5.53	5.73	149.91	4.19	2.80
6	87.03	5.22	6.00	148.76	3.28	2.20

Overall the hybrid architecture is more robust. It combines lower CV values, smaller spreads and the absence of outliers ensuring that its higher average throughput is also achieved with greater consistency and reliability.

8.6. Summary of Key Findings

In summary, this comparative analysis shows that hybrid architectures deliver higher throughput, scalability, efficiency and robustness, especially as warehouse size and shuttle density increase. The hybrid model has the ability to avoid performance saturation, maintain efficient resource use and deliver consistent results across a wide operational envelope positions. This makes it as a strong candidate for I4.0-ready, large-scale SBSRS deployments. Table 8.2 summarizes the simulation results across the four KPIs.

Table 8.2: Quantitative and Qualitative Summary of Results Across Control Architectures

Aspect	Hierarchical Architecture	Hybrid Architecture	Net Effect
Throughput	Peaks early then declines: 98 → 70 tasks/hour from 2 to 6 shuttles due to congestion.	Scales effectively with more shuttles: peaks at 128 tasks/hour (5 shuttles), remains high at 6.	Hybrid achieves 30-60% higher throughput in large layouts; up to +52 tasks/hour (+90%) in extreme cases.
Scalability	Throughput halves from 152 to 77 tasks/hour when scaling from 10×10 to 25×25 (3 shuttles); limited by coordination bottlenecks.	Throughput declines more gradually: 186 to 109 tasks/hour; performs well even in large systems.	Hybrid scales reliably without saturation; avoids bottlenecks seen in centralized control.
Operational Efficiency	Normalized throughput drops from 8.1 to 1.1 tasks/hour/unit distance as congestion increases; efficiency loss due to idle travel.	Declines from 8.6 to 1.7 tasks/hour/unit distance, but more gradually; efficient routing reduces waste.	Hybrid maintains up to +68% higher movement efficiency in dense scenarios; less wasted motion.
Robustness	IQR: 7.4-8.0 tasks/hour; CV: 2.7-6.0%; 1-2 outliers per scenario with 3+ shuttles, some $<$ 90 tasks/hour.	IQR: 6.8-7.1 tasks/hour; CV: 2.2-2.8%; no outliers across all cases.	Similar central variability, but hybrid eliminates outliers and reduces CV by up to 3%.
Behavioral Insights	Centralized FIFO leads to sequential aisle use, blocking, and frequent deadlocks with 3+ shuttles.	Agents negotiate path access and reroute dynamically, dispersing congestion before deadlocks emerge.	Hybrid enables adaptive, scalable coordination aligned with I4.0 principles; hierarchical control becomes rigid under load.

8.7. Discussion

This section discusses the findings by interpreting the results and their broader relevance while also addressing the main limitations of the study.

8.7.1. Interpretation of Results

The simulation experiments clearly show that the choice of control architecture has a major impact on SBSRS performance. Across all scenarios the hybrid MAS architecture consistently outperforms the hierarchical baseline although the reasons for these gains differ across the various KPIs.

For throughput, hierarchical control peaks at low shuttle counts and quickly levels off. This comes from its centralized scheduling approach where global path reservations and sequential task assignment create bottlenecks once multiple shuttles compete for resources. After a certain point adding more shuttles no longer increases capacity since congestion and blocking dominate system behavior. In contrast, hybrid control continues to benefit from additional shuttles. Local routing and distributed task allocation allow shuttles to operate in parallel, which explains why throughput keeps improving up to five or six shuttles before stabilizing [49].

When scaling system size both architectures show reduced performance due to longer travel distances but the drop is steeper in the hierarchical model. Centralized coordination struggles with the growing complexity of larger networks where global planning becomes rigid and prone to bottlenecks. Hybrid control sustains higher throughput at all sizes even though absolute performance still decreases in bigger systems. This means hybrid scalability comes not from avoiding performance loss completely but from keeping throughput at a consistently higher level as warehouses grow [69].

Operational efficiency highlights another difference. Hierarchical control suffers from wasted travel and idle time as shuttles often wait for entire path reservations to clear. This inefficiency grows in larger and denser systems where more shuttles compete for limited cross-aisle capacity. Hybrid control reduces this wasted effort by letting shuttles adapt their routes to local conditions and share workload more effectively. As a result shuttle movements translate more directly into productive work explaining the clear efficiency advantage seen in the normalized throughput results [69].

Robustness further illustrates the stability of hybrid performance. While both models show similar central spreads, the hierarchical system often produces outliers including runs with much lower throughput due to congestion spirals or poor task–shuttle matches. Hybrid control avoids these low-performing cases and keeps variability lower overall leading to more predictable system output. In practice, this reliability is as important as high average performance since industrial operations depend on stability and worst-case guarantees [59].

Two broader observations also stand out. First, in small and simple systems with only a few shuttles, the performance differences between hierarchical and hybrid control remain limited. This suggests that centralized control may still be suitable in low-demand settings while the benefits of hybrid control become more important as systems scale in size and complexity. Second, the higher movement efficiency of hybrid control points to possible long-term benefits such as lower energy consumption and reduced mechanical wear. These factors were not modeled directly but they indicate advantages that go beyond throughput alone.

Overall, the results show that the performance gap between hierarchical and hybrid control is not caused by isolated factors but by fundamental architectural differences. Hierarchical control is limited by its centralized nature which restricts scalability, increases inefficiency and exposes the system to variability under load. Hybrid MAS, by contrast, uses distributed autonomy and local coordination to maintain throughput, efficiency and stability across a wide range of conditions. This makes hybrid architectures better suited to the dynamic and high-throughput demands of modern SBSRS and aligns them closely with the goals of I4.0.

While the performance advantages of hybrid MAS are clear across all tested scenarios, the extent

to which these results can be generalized must be considered carefully. The results apply most directly to SBSRS with layouts, task patterns and operating rules similar to those modeled here. Still, the architectural mechanisms driving the differences between hybrid and hierarchical control like congestion management, distributed decision-making and robustness are not unique to the simulated system. These are structural properties of control architectures that would influence performance in a wide range of SBSRS configurations.

This means that even though the exact throughput values may vary with different warehouse geometries, demand profiles or multi-level system designs, the relative trends are likely to remain the same. Hierarchical control struggles with scalability and variability while hybrid maintains higher throughput, efficiency and stability. The generalizability of the findings is therefore rooted less in the specific case parameters and more in the demonstration of how architectural design fundamentally shapes performance dynamics across scales. In that sense, this study provides insights that extend beyond the particular system modeled offering lessons applicable to other SBSRS layouts and, more broadly, to automated material-handling systems that rely on fleets of cooperating vehicles.

8.7.2. Limitations

While this simulation-based comparison provides robust evidence on the performance of hierarchical and hybrid MAS architectures, the following limitations should be considered. Where possible the expected impact or rationale is discussed.

Model and Experimental Simplifications

- **Idealized Operation:** The model assumes flawless operations: no equipment failures, no task interruptions and perfect shuttle execution. Real-world disturbances such as breakdowns, delays or cancellations were not included. This simplification was made to isolate architectural effects under controlled conditions. It may overestimate throughput and hide some operational bottlenecks. Importantly, literature suggests that hybrid MAS would perform even better in disturbed conditions as distributed control is naturally more fault-tolerant [59].
- **Simplified Shuttle Dynamics:** Shuttle movements are modeled using fixed travel times without acceleration, deceleration or turning delays. While this reduces realism, it ensures that results reflect only the effect of control logic rather than physical motion detail. In real systems more detailed dynamics could influence congestion patterns but relative differences between architectures are unlikely to change. Simplification also improves computational efficiency.
- **Single-Level System:** Only a single-level SBSRS was modeled without lifts or vertical transfers. This restriction reflects common real-world implementations but excludes scenarios where multi-level coordination is critical. In such systems hybrid MAS is likely to have an even stronger advantage due to its ability to coordinate multiple agent types (e.g., shuttles and lifts).

Control Logic and Communication

- **Perfect Communication:** All agent communication is assumed to be instantaneous and error-free. In practice, network latency or message loss could affect coordination especially in large distributed systems. While this might reduce hybrid responsiveness under heavy network load, practical mitigation strategies (e.g., local fallback rules) exist. The assumption was made to avoid technology-specific bias and focus on core architectural differences.
- **Abstracted Control Logic:** The hierarchical and hybrid MAS models are conceptual rather than industrial implementations. The hierarchical system is based on simplified industrial practices (e.g., Vanderlande's ADAPTO), while the hybrid is built on frameworks from literature. This abstraction excludes low-level detail but ensures transparency and fair comparison between the two models.

Validation, Generalizability, and Scope

- **No Real-World Validation:** The models were not calibrated or validated with operational SBSRS data which limits external validity. Results should therefore be seen as conceptual insights, not deployment-ready recommendations. Calibration against industrial data or development of a digital twin would be logical next steps.

- **No Cost-Benefit Analysis:** The study focuses only on technical KPIs (throughput, efficiency, scalability, robustness). It does not assess costs. In practice, hybrid MAS may involve additional expenses (hardware, integration, agent programming). For adoption, these must be justified by sufficient performance gains.
- **Limited System Layouts:** The scenarios reflect a specific SBSRS configuration (single cross-aisle, tier-captive). Other layouts, traffic rules or load policies may lead to different outcomes. The findings are therefore most relevant to systems with similar characteristics.
- **Statistical and Computational Limits:** Run lengths and replication counts were set to balance statistical validity with computational feasibility. While sufficient for robust results, rare or extreme behaviors (e.g., very high congestion) may not have been fully captured.

In summary, while the findings provide reliable comparative insights, they cannot be generalized to all SBSRS designs or directly applied to industry without further validation. Most simplifications were deliberate choices to keep the comparison fair and computationally feasible. Future work should address these limitations to better connect conceptual results with practical applications.

9

Conclusion & Recommendations

This study set out to answer the main research question: *What is the impact of control architectures on the operational performance of a Shuttle-Based Storage and Retrieval System?* To address this, a comparative simulation study was conducted using a simplified case of the Vanderlande ADAPTO system. The two control architectures were designed and implemented in a combined DES-ABM framework: a conventional hierarchical model and an I4.0-inspired hybrid control architecture. Both were tested and compared across a wide range of warehouse sizes and shuttle densities. System performance was evaluated based on four key performance indicators: throughput, efficiency, scalability and robustness.

The simulation results demonstrate that control architecture selection strongly affects SBSRS performance. The hybrid model consistently outperforms the hierarchical architecture across all performance metrics and scenarios. The performance gap was most pronounced in large-scale environments operating with a high number of concurrent shuttles. The main findings can be summarized as follows:

- Throughput: Up to 90% higher in large-scale, high-density systems; 30-60% higher in mid-sized configurations; up to 36% higher under optimal comparable conditions.
- Efficiency: Up to 68% improvement in normalized throughput (tasks per distance unit).
- Scalability: Sustained or improving performance as system size and shuttle count increased, unlike hierarchical control which often degraded beyond two or three shuttles.
- Robustness: Significantly lower variance across replications, with hybrid control eliminating the low-outlier runs seen under centralized coordination.

In conclusion, this research shows that the way control is structured in an SBSRS has a significant impact on its operational performance. Using an ADAPTO-inspired case study, the results demonstrate that the hybrid model is more scalable, more efficient and more robust. The performance gap is most evident in large and highly dynamic systems where hierarchical control struggled with congestion and coordination demands. Hybrid MAS control provides a viable and future-proof alternative that not only improves performance but also supports the integration of Industry 4.0 technologies such as local autonomy, intelligent routing and real-time communication. The key takeaway from this work is therefore the following: the control architecture itself is a critical design decision in SBSRS and hybrid control architectures offer a clear technical advantage and a strong basis for future I4.0 integration.

9.1. Contribution

This research makes several contributions to both theory and practice. On the academic side, it addresses two main gaps in existing literature. First, there are very few studies that directly compare different control architectures in SBSRS. Second, current SBSRS research rarely includes core Industry 4.0 concepts such as local autonomy and agent-based coordination [49, 19]. Although these ideas have been discussed in broader logistics research, there has not been a systematic, quantitative comparison within the context of SBSRS. This thesis tackles that gap by using simulation to compare hierarchical and hybrid control architectures, and by showing how control structure affects four system performance metrics, throughput, scalability, efficiency and robustness across different system

sizes and shuttle densities. In doing so, it strengthens understanding of how control architectures influence the operational behavior of SBSRS and provides a quantitative, simulation-based comparison of hierarchical and hybrid MAS in this context. More broadly, the results support and extend existing expectations in the literature, showing that centralized control has clear limitations and that partial local agent autonomy can improve performance in dynamic material handling systems.

From a practical point of view, the results offer useful guidance for warehouse designers and automation engineers. The findings show that hybrid MAS control can deliver clear improvements in throughput especially in large or high-density systems where centralized control becomes a bottleneck. By focusing on shuttle-level autonomy and combining it with supervisory control, the proposed hybrid architecture can also be applied in real-world SBSRS without the need to completely rebuild the existing system architecture.

9.2. Recommendations for Future Work

Future research should extend the model and its scope. In particular, the simulation could be expanded to include multi-level SBSRS layouts with vertical lifts, buffering zones and more complex routing layers. The literature suggests that these system extensions could even increase the benefits of distributed control, since agent-based approaches are better suited to managing interdependent tasks and coordinating both horizontal and vertical movements. Another useful direction would be to include stochastic elements, such as equipment failures, communication delays and task interruptions in order to test how well the hybrid MAS performs under imperfect conditions. Earlier studies have shown that agent-based systems often outperform centralized control in the presence of disturbances, so it is likely that the advantages of the hybrid architecture would be even more pronounced in such cases.

Another important direction for future work is to further develop the hybrid MAS architecture so that it can fully support the integration of Industry 4.0 technologies. One of the goals of this research was not only to improve performance but also to prepare the SBSRS control architecture for I4.0 capabilities. The current hybrid model provides this foundation and opens doors for additional I4.0 integration. Future research could therefore focus on optimizing the hybrid architecture for example by exploring advanced agent collaboration mechanisms, dynamic negotiation protocols and the impact of communication delays or network imperfections. In addition, AI and machine-learning techniques could be used to support features such as adaptive path planning, real-time learning and dynamic task allocation. These additions would improve the responsiveness and flexibility of the system and would help the architecture to achieve full I4.0 potential, one in which shuttles act as smart, interconnected agents capable of learning and adapting in real time.

Next to scope extension and I4.0 integration, there are other recommendations to build on this research. Empirical validation of the hybrid MAS approach is recommended, either through detailed full-system simulations or phased pilot implementations in controlled environments. This would help bridge the gap between conceptual findings and industrial application. Next to that, it would be valuable to assess the technical feasibility of deploying such architectures by examining the mechanical and digital capabilities required of shuttle agents. This includes real-time positioning, onboard sensing, RFID-based task identification and communication via IoT infrastructure, all of which are essential for operationalizing the distributed decision-making demonstrated in this study.

Next to technical performance, future studies should also include a comprehensive cost-benefit analysis that evaluates capital expenditures, operational expenses and return on investment. This is particularly important for industrial stakeholders (such as Vanderlande Industries) as the ultimate objective of any automated system is not only to perform well but to do so in a cost-efficient way. While this study demonstrates the technical value of hybrid control, the economic value still needs to be assessed. Determining whether the technical gains are large enough to justify the additional complexity required for hybrid control would be highly relevant for any future industrial application.

Bibliography

[1] Azadeh, K., De Koster, R., and Roy, D. “Robotized and automated warehouse systems: Review and recent developments”. In: *Transportation Science* 53.4 (2019), pp. 917–945.

[2] Azadeh, K., De Koster, R., and Roy, D. “Robotized warehouse systems: Developments and research opportunities”. In: *ERIM report series research in management Erasmus Research Institute of Management* ERS-2017-009-LIS (2017).

[3] Azadeh, K., Roy, D., and De Koster, R. “Design, modeling, and analysis of vertical robotic storage and retrieval systems”. In: *Transportation Science* 53.5 (2019), pp. 1213–1234.

[4] Balci, O. “Validation, verification, and testing techniques throughout the life cycle of a simulation study”. In: *Annals of operations research* 53 (1994), pp. 121–173.

[5] Banks, J. *Discrete event system simulation*. Pearson Education India, 2005.

[6] Basile, F., Chiacchio, P., and Di Marino, E. “An auction-based approach for the coordination of vehicles in automated warehouse systems”. In: *2017 IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI)*. IEEE. 2017, pp. 121–126.

[7] Bonabeau, E. “Agent-based modeling: Methods and techniques for simulating human systems”. In: *Proceedings of the national academy of sciences* 99.suppl_3 (2002), pp. 7280–7287.

[8] Boysen, N., De Koster, R., and Weidinger, F. “Warehousing in the e-commerce era: A survey”. In: *European Journal of Operational Research* 277.2 (2019), pp. 396–411.

[9] Boysen, N., Koster, R. de, and Füßler, D. “The forgotten sons: Warehousing systems for brick-and-mortar retail chains”. In: *European Journal of Operational Research* 288.2 (2021), pp. 361–381.

[10] Carlo, H. J. and Vis, I. F. “Sequencing dynamic storage systems with multiple lifts and shuttles”. In: *International Journal of Production Economics* 140.2 (2012), pp. 844–853.

[11] Custodio, L. and Machado, R. “Flexible automated warehouse: a literature review and an innovative framework”. In: *The International Journal of Advanced Manufacturing Technology* 106 (2020), pp. 533–558.

[12] De Koster, R., Le-Duc, T., and Roodbergen, K. J. “Design and control of warehouse order picking: A literature review”. In: *European journal of operational research* 182.2 (2007), pp. 481–501.

[13] Deng, L., Chen, L., Zhao, J., and Wang, R. “Modeling and performance analysis of shuttle-based compact storage systems under parallel processing policy”. In: *Plos one* 16.11 (2021), e0259773.

[14] Derigent, W., Cardin, O., and Trentesaux, D. “Industry 4.0: contributions of holonic manufacturing control architectures and future challenges”. In: *Journal of Intelligent Manufacturing* 32.7 (2021), pp. 1797–1818.

[15] Dotoli, M. and Fanti, M. P. “A coloured Petri net model for automated storage and retrieval systems serviced by rail-guided vehicles: a control perspective”. In: *International Journal of Computer Integrated Manufacturing* 18.2-3 (2005), pp. 122–136.

[16] Eder, M. “An analytical approach of multiple-aisle shuttle-based storage and retrieval systems”. In: *The International Journal of Advanced Manufacturing Technology* 127.3 (2023), pp. 1585–1596.

[17] Ekren, B. Y. “Transaction Processing Policies in a Flexible Shuttle-based Storage and Retrieval System by Real-time Data Tracking under Agent-based Modelling”. In: (2023).

[18] Ekren, B. Y., Heragu, S. S., Krishnamurthy, A., and Malmborg, C. J. “An approximate solution for semi-open queueing network model of an autonomous vehicle storage and retrieval system”. In: *IEEE Transactions on Automation Science and Engineering* 10.1 (2012), pp. 205–215.

[19] Ekren, B. Y., Kaya, B., and Küçükyaşar, M. “Shuttle-based storage and retrieval systems designs from multi-objective perspectives: total investment cost, throughput rate and sustainability”. In: *Sustainability* 15.1 (2022), p. 762.

[20] Eroglu Turhanlar, E., Yetkin Ekren, B., and Lerher, T. “Autonomous mobile robot travel under deadlock and collision prevention algorithms by agent-based modelling in warehouses”. In: (2022).

[21] Fan, H. et al. “A Lifelong Multi-Shuttle Scheduling Framework for the AS/RS System”. In: *IEEE Transactions on Automation Science and Engineering* (2025).

[22] Félix-Cigalat, J. S. and Domingo, R. “Towards a digital twin warehouse through the optimization of internal transport”. In: *Applied Sciences* 13.8 (2023), p. 4652.

[23] Ferrari, A., Zenezini, G., Rafele, C., and Carlin, A. “A Roadmap towards an Automated Warehouse Digital Twin: current implementations and future developments”. In: *IFAC-PapersOnLine* 55.10 (2022), pp. 1899–1905.

[24] Fischer, J., Lieberoth-Leden, C., Fottner, J., and Vogel-Heuser, B. “Design, application, and evaluation of a multiagent system in the logistics domain”. In: *IEEE Transactions on Automation Science and Engineering* 17.3 (2020), pp. 1283–1296.

[25] Fishman, G. S. *Discrete-event simulation: modeling, programming, and analysis*. Vol. 537. Springer, 2001.

[26] Folgado, F. J., Calderón, D., González, I., and Calderón, A. J. “Review of Industry 4.0 from the perspective of automation and supervision systems: Definitions, architectures and recent trends”. In: *Electronics* 13.4 (2024), p. 782.

[27] Fottner, J. et al. “Autonomous systems in intralogistics: State of the art and future research challenges”. In: *Logistics Research* 14.1 (2021), pp. 1–41.

[28] Fragapane, G., De Koster, R., Sgarbossa, F., and Strandhagen, J. O. “Planning and control of autonomous mobile robots for intralogistics: Literature review and research agenda”. In: *European journal of operational research* 294.2 (2021), pp. 405–426.

[29] Frazelle, E. H. *World-class warehousing and material handling*. McGraw-Hill Education, 2016.

[30] Gademann, N. and Velde, S. “Order batching to minimize total travel time in a parallel-aisle warehouse”. In: *IIE transactions* 37.1 (2005), pp. 63–75.

[31] Gagliardi, J.-P., Renaud, J., and Ruiz, A. “Models for automated storage and retrieval systems: a literature review”. In: *International Journal of Production Research* 50.24 (2012), pp. 7110–7125.

[32] Gomez, C., Guardia, A., Mantari, J., Coronado, A., and Reddy, J. “A contemporary approach to the MSE paradigm powered by artificial intelligence from a review focused on polymer matrix composites”. In: *Mechanics of Advanced Materials and Structures* 29.21 (2022), pp. 3076–3096.

[33] Gu, J., Goetschalckx, M., and McGinnis, L. F. “Research on warehouse operation: A comprehensive review”. In: *European journal of operational research* 177.1 (2007), pp. 1–21.

[34] Güller, M. and Hegmanns, T. “Simulation-based performance analysis of a miniload multishuttle order picking system”. In: *Procedia CIRP* 17 (2014), pp. 475–480.

[35] Gutelius, B. and Theodore, N. “Pandemic-Related Trends in Warehouse Technology Adoption”. In: (2023).

[36] Ha, Y. and Chae, J. “Free balancing for a shuttle-based storage and retrieval system”. In: *Simulation Modelling Practice and Theory* 82 (2018), pp. 12–31.

[37] Helo, P. and Hao, Y. “Artificial intelligence in operations management and supply chain management: An exploratory case study”. In: *Production planning & control* 33.16 (2022), pp. 1573–1590.

[38] Hu, J. et al. “To centralize or not to centralize: A tale of swarm coordination”. In: *arXiv preprint arXiv:1805.01786* (2018).

[39] Ikumapayi, O. M., Laseinde, O. T., Elewa, R. R., Ogedengbe, T. S., and Akinlabi, E. T. “Swarm Robotics in a Sustainable Warehouse Automation: Opportunities, Challenges and Solutions”. In: *E3S Web of Conferences*. Vol. 552. EDP Sciences. 2024, p. 01080.

[40] Ilona, H. *Shuttle-based Storage and Retrieval System*. Image posted on LinkedIn by a Vanderlande employee. 2025. URL: https://www.linkedin.com/posts/illonahoe%20nderop_last-week-we-traveled-to-siegen-to-align-ugcPost-7289349467%20718254%20592-s10c?utm_source=share&utm_medium=member_desktop&rcm=ACoAADBZip%20QBEhJieBUSSDYS4HpSrcW3Cu8vxAM (visited on 08/15/2025).

[41] Jennings, N. R., Sycara, K., and Wooldridge, M. "A roadmap of agent research and development". In: *Autonomous agents and multi-agent systems* 1.1 (1998), pp. 7–38.

[42] Kaczmarek, S., Goldenstein, J., and Hompel, M. ten. "Performance analysis of autonomous vehicle storage and retrieval systems depending on storage management policies". In: *2014 IEEE International Conference on Industrial Engineering and Engineering Management*. IEEE. 2014, pp. 1424–1428.

[43] Kagermann, H., Wahlster, W., and Helbig, J. "Securing the future of German manufacturing industry: Recommendations for implementing the strategic initiative INDUSTRIE 4.0". In: *Final report of the Industrie* 4.0 (2013).

[44] Kattepur, A., Rath, H., Mukherjee, A., and Simha, A. "Distributed optimization framework for industry 4.0 automated warehouses". In: *EAI Endorsed Transactions on Industrial Networks and Intelligent Systems* 5.15 (2018).

[45] Koster, R. de. "Warehousing 2030". In: *Global Logistics and Supply Chain Strategies for the 2020s: Vital Skills for the Next Generation* (2022), pp. 243–260.

[46] Kuhl, M. E., Bhisti, R., Bhattathiri, S. S., and Li, M. P. "Warehouse Digital Twin: Simulation Modeling and Analysis Techniques". In: *2022 Winter Simulation Conference (WSC)*. IEEE. 2022, pp. 2947–2956.

[47] Law, A. M. *Simulation modeling and analysis*. Vol. 5. Mcgraw-hill Education, 2015.

[48] Law, A. M., Kelton, W. D., and Kelton, W. D. *Simulation modeling and analysis*. Vol. 3. Mcgraw-hill New York, 2000.

[49] Leitão, P. "Agent-based distributed manufacturing control: A state-of-the-art survey". In: *Engineering applications of artificial intelligence* 22.7 (2009), pp. 979–991.

[50] Leitão, P. and Karnouskos, S. "Industrial agents: emerging applications of software agents in industry". In: (2015).

[51] Lepuschitz, W. "Self-reconfigurable manufacturing control based on ontology-driven automation agents". PhD thesis. Technische Universität Wien, 2018.

[52] Lerher, T., Yetkin Ekren, B., Sari, Z., and Rosi, B. "Simulation analysis of shuttle based storage and retrieval systems". In: *Internatıonal Journal of Sımulatıon Modellıng* (2015).

[53] Li, Y. and Li, Z. "Shuttle-Based Storage and Retrieval System: A Literature Review". In: *Sustainability* 14.21 (2022), p. 14347.

[54] Liu, X., Cao, J., Yang, Y., and Jiang, S. "CPS-based smart warehouse for industry 4.0: A survey of the underlying technologies". In: *Computers* 7.1 (2018), p. 13.

[55] Macal, C. M. and North, M. J. "Tutorial on agent-based modeling and simulation". In: *Proceedings of the Winter Simulation Conference, 2005.* IEEE. 2005, 14–pp.

[56] Mao, J., Cheng, J., Li, X., Zhao, H., and Lin, C. "Modelling analysis of a four-way shuttle-based storage and retrieval system on the basis of operation strategy". In: *Applied Sciences* 13.5 (2023), p. 3306.

[57] Marik, V. and McFarlane, D. "Industrial adoption of agent-based technologies". In: *IEEE intelligent systems* 20.1 (2005), pp. 27–35.

[58] Moneva, H., Caarls, J., and Verriet, J. "A holonic approach to warehouse control". In: *7th International Conference on Practical Applications of Agents and Multi-Agent Systems (PAAMS 2009)*. Springer. 2009, pp. 1–10.

[59] Monostori, L. et al. "Cyber-physical systems in manufacturing". In: *Cirp Annals* 65.2 (2016), pp. 621–641.

[60] Murata, T. “Petri nets: Properties, analysis and applications”. In: *Proceedings of the IEEE* 77.4 (2002), pp. 541–580.

[61] Nagy, G., Illés, B., and Bányai, Á. “Impact of Industry 4.0 on production logistics”. In: *IOP conference series: Materials science and engineering*. Vol. 448. 1. IOP Publishing. 2018, p. 012013.

[62] Oks, S. J. et al. “Cyber-physical systems in the context of industry 4.0: A review, categorization and outlook”. In: *Information Systems Frontiers* 26.5 (2024), pp. 1731–1772.

[63] Pagone, E., Haddad, Y., Barsotti, L., Dini, G., and Salonitis, K. “A stochastic evaluation framework to improve the robustness of manufacturing systems”. In: *International Journal of Computer Integrated Manufacturing* 36.7 (2023), pp. 966–984.

[64] Reaidy, P. J., Gunasekaran, A., and Spalanzani, A. “Bottom-up approach based on Internet of Things for order fulfillment in a collaborative warehousing environment”. In: *International Journal of Production Economics* 159 (2015), pp. 29–40.

[65] Robinson, S. *Simulation: the practice of model development and use*. Bloomsbury Publishing, 2014.

[66] Rocca, R., Rosa, P., Sassanelli, C., Fumagalli, L., and Terzi, S. “Integrating virtual reality and digital twin in circular economy practices: A laboratory application case”. In: *Sustainability* 12.6 (2020), p. 2286.

[67] Roodbergen, K. J. and Vis, I. F. “A survey of literature on automated storage and retrieval systems”. In: *European journal of operational research* 194.2 (2009), pp. 343–362.

[68] Rossetti, M. D. *Simulation Modeling and Arena*. 3rd and Open Text Edition. Licensed under CC BY-NC-ND 4.0. Open Access, 2021. URL: `https://rossetti.github.io/RossettiArenaBook/`.

[69] Salvador Palau, A., Dhada, M. H., and Parlikad, A. K. “Multi-agent system architectures for collaborative prognostics”. In: *Journal of Intelligent Manufacturing* 30.8 (2019), pp. 2999–3013.

[70] Sargent, R. “Verification and validation of simulation models”. In: *Journal of Simulation* (2012).

[71] Sargent, R. G. “Statistical analysis of simulation output data”. In: *ACM SIGSIM Simulation Digest* 7.4 (1976), pp. 39–50.

[72] Sterman, J. D. et al. “Systems thinking and modeling for a complex world”. In: *Management* 6.1 (2000), pp. 7–17.

[73] Tako, A. A. and Robinson, S. “The application of discrete event simulation and system dynamics in the logistics and supply chain context”. In: *Decision support systems* 52.4 (2012), pp. 802–815.

[74] Tikwayo, L. N. and Mathaba, T. N. “Applications of industry 4.0 technologies in warehouse management: A systematic literature review”. In: *Logistics* 7.2 (2023), p. 24.

[75] Tubis, A. A. and Rohman, J. “Intelligent warehouse in industry 4.0—systematic literature review”. In: *Sensors* 23.8 (2023), p. 4105.

[76] Van Brussel, H., Wyns, J., Valckenaers, P., Bongaerts, L., and Peeters, P. “Reference architecture for holonic manufacturing systems: PROSA”. In: *Computers in industry* 37.3 (1998), pp. 255–274.

[77] Van Den Berg, J. P. “A literature survey on planning and control of warehousing systems”. In: *IIE transactions* 31.8 (1999), pp. 751–762.

[78] Van Den Berg, J. P. and Gademann, A. “Optimal routing in an automated storage/retrieval system with dedicated storage”. In: *IIE transactions* 31.5 (1999), pp. 407–415.

[79] Van den Berg, J. P. and Zijm, W. H. “Models for warehouse management: Classification and examples”. In: *International journal of production economics* 59.1-3 (1999), pp. 519–528.

[80] Van Gils, T., Caris, A., Ramaekers, K., and Braekers, K. “Formulating and solving the integrated batching, routing, and picker scheduling problem in a real-life spare parts warehouse”. In: *European Journal of Operational Research* 277.3 (2019), pp. 814–830.

[81] Wang, Y., Mou, S., and Wu, Y. “Storage assignment optimization in a multi-tier shuttle warehousing system”. In: *Chinese journal of mechanical engineering* 29.2 (2016), pp. 421–429.

[82] Wang, Y., Mou, S., and Wu, Y. “Task scheduling for multi-tier shuttle warehousing systems”. In: *International Journal of Production Research* 53.19 (2015), pp. 5884–5895.

[83] Welch, P. D. “The statistical analysis of simulation results”. In: *Computer performance modeling handbook* (1983).

[84] Wilensky, U. and Rand, W. *An introduction to agent-based modeling: modeling natural, social, and engineered complex systems with NetLogo*. MIT press, 2015.

[85] Winkelhaus, S. and Grosse, E. H. “Logistics 4.0: a systematic review towards a new logistics system”. In: *International journal of production research* 58.1 (2020), pp. 18–43.

[86] Wooldridge, M. *An introduction to multiagent systems*. John wiley & sons, 2009.

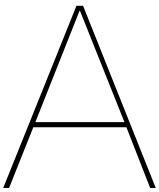
[87] Xu, X., Zou, B., Shen, G., and Gong, Y. “Travel-time models and fill-grade factor analysis for double-deep multi-aisle AS/RSs”. In: *International Journal of Production Research* 54.14 (2016), pp. 4126–4144.

[88] Yang, P., Miao, L., Xue, Z., and Qin, L. “An integrated optimization of location assignment and storage/retrieval scheduling in multi-shuttle automated storage/retrieval systems”. In: *Journal of Intelligent Manufacturing* 26 (2015), pp. 1145–1159.

[89] Youssef, A. A., El Khoreby, M. A., Issa, H. H., and Abdellatif, A. “Brief survey on Industry 4.0 warehouse management systems”. In: *International Review on Modelling and Simulations* 15.5 (2022), pp. 340–350.

[90] ZHANG, R., Zhuan, W., et al. “Research on Multi-objective Dynamic Task Scheduling of Cross-aisles Multi-layer Shuttle Truck Storage System Based on Elite Strategy”. In: *MATEC Web of Conferences*. Vol. 325. EDP Sciences. 2020, p. 05002.

[91] Zhou, K., Liu, T., and Zhou, L. “Industry 4.0: Towards future industrial opportunities and challenges”. In: *2015 12th International conference on fuzzy systems and knowledge discovery (FSKD)*. IEEE. 2015, pp. 2147–2152.



Scientific Research Paper

The Impact of Control Architectures on the Performance of Shuttle-Based Storage and Retrieval Systems in Industry 4.0 Environments: A Simulation-Based Comparison of Hierarchical and Hybrid Architectures

C.S.E. van Oeveren[a], Dr. ir. Y. Pang[a], H. Huijsman[b], and Prof. dr. R.R. Negenborn[a]

[a] *Faculty of Mechanical Engineering, Delft University of Technology, Delft, The Netherlands*
[b] *Vanderlande Industries, Veghel, The Netherlands*

***Abstract*— Industry 4.0 technologies have the potential to transform logistics by enabling smarter, more autonomous warehouse systems. However, Shuttle-Based Storage and Retrieval Systems (SBSRS) still rely largely on centralized or hierarchical control architectures, limiting their adaptability, scalability and responsiveness. While existing research has focused on optimizing control policies within these conventional frameworks, the architectural layer itself remains underexplored. This study investigates the impact of control architectures on SBSRS performance by comparing a conventional hierarchical model with a hybrid control architecture that introduces local autonomy and inter-shuttle communication. A hybrid Discrete-Event and Agent-Based Simulation model was developed to evaluate both architectures, using a tier-captive SBSRS inspired by Vanderlande's ADAPTO system. System Performance was assessed under varying system sizes and shuttle densities using throughput, normalized efficiency, scalability and robustness as key performance indicators. The results show that the hybrid architecture consistently outperforms the centralized approach in throughput, especially in large scale system, with up to 36% throughput gain, up to 68% efficiency gain (normalized throughput), improved scalability and reduced variability. These findings suggest that hybrid control offers a promising path for designing adaptive, Industry 4.0-ready SBSRS. The study contributes a conceptual design and simulation-based comparison to guide future control architecture development.**

***Keywords*—Shuttle-Based Storage and Retrieval Systems (SBSRS), control architectures, Industry 4.0 (I4.0), Multi-Agent Systems (MAS), hybrid control, simulation, smart warehousing**

I. Introduction

The growing demand for high-speed, high-volume order fulfillment has positioned warehouses as critical nodes in modern supply chains. Shuttle-Based Storage and Retrieval Systems (SBSRS), a type of automated warehousing technology, are widely used for their ability to deliver high throughput and space efficiency. These systems rely on fleets of autonomous shuttles operating on fixed rails to perform storage and retrieval tasks in parallel. This makes them ideal for dynamic environments such as e-commerce fulfillment centers and automated warehouses [1]. SBSRS rely on simultaneous shuttle operations, making the system's control architecture a key determinant of overall performance, particularly in coordinating resources and in handling congestion.

Conventionally, control systems in SBSRS rely on centralized or hierarchical architectures, where a central or higher-level control unit dictates, among others, task assignments, path planning and congestion resolution. While such designs facilitate global optimization, they create performance bottlenecks, limit responsiveness and pose risks of single-point failure [6, 12]. In large-scale systems these limitations can significantly degrade performance due to the system's dynamic nature, parallelism and the need for responsive decision-making across multiple shuttles [17, 15].

As warehousing systems evolve under Industry 4.0 (I4.0), these architectural limitations become increasingly evident. I4.0 technologies such as the Internet of Things (IoT), Artificial Intelligence (AI) and Cyber-Physical Systems (CPS) introduce key capabilities including distributed control, real-time communication and local autonomy. These capabilities conflict with the rigid, top-down nature of conventional SBSRS control. Integrating I4.0 capabilities into SBSRS could unlock greater responsiveness and adaptability and can transform the storage and retrieval operations from static, top-down systems into adaptive, intelligent environments [21].

Within the field of SBSRS existing research on system improvement has focused on optimizing control policies within existing centralized or hierarchical frameworks, the architectural layer itself remains underexplored [11, 14]. As a result, there is limited understanding of how architectural redesign especially toward distributed and agent-based paradigms can affect SBSRS performance.

This paper addresses this gap by evaluating how different control architectures impact SBSRS performance in environments enabled by Industry 4.0 conditions. The research is guided by the following question: *What is the impact of control architectures on the operational performance of a Shuttle-Based Storage and Retrieval System?*

The key contributions of this work are as follows. First, it presents a conceptual and operational design of two contrasting SBSRS control architectures: a centralized hierarchical model and a hybrid model using MAS technology. Second, a DES-ABM simulation framework was developed to support both distributed agent-based logic and traditional centralized control, enabling direct comparison within a unified environment. Third, the study conducts a systematic performance evaluation across varying system sizes and shuttle densities using key KPIs including throughput, normalized efficiency, scalability and robustness. Fourth, it provides quantitative evidence that hybrid MAS control improves throughput by up to 36% and efficiency by up to 68% in large-scale settings and exhibits higher scalability and robustness. Finally, the work offers a practical foundation for incorporating Industry 4.0 principles like autonomy, coordination and adaptability into future SBSRS control designs.

This study focuses on the shuttle agent, as shuttle coordination is central to SBSRS performance and congestion dynamics. The system abstraction is based on a tier-captive SBSRS inspired by Vanderlande's ADAPTO system. By concentrating on a single-level layout and excluding lifts, buffers and failure events, the simulation isolates architectural effects and enables targeted evaluation of control logic.

II. Background

This section outlines the system context, control architecture designs and relevant research on SBSRS, forming the foundation for the study's architectural focus.

A. SBSRS System Analysis

Shuttle-Based Storage and Retrieval Systems are commonly used in high-throughput environments such as those in retail, e-commerce and manufacturing. These systems consist of modular racks, tier-captive shuttles that travel along fixed rails and vertical lifts to support vertical movement. SBSRS provide excellent scalability and space utilization through their ability to operate multiple shuttles in parallel on different tiers [8].

The performance of SBSRS depends not only on their mechanical configuration but also on their control logic and architecture. Typical control functions include storage assignment, task allocation, routing, deadlock prevention and congestion handling. According to current academic and industrial designs these functions predominantly rely on centralized or hierarchical control architectures [6], where a Warehouse Control System (WCS) or higher-level controller issues top-down instructions to shuttle agents and manages global coordination [12, 17, 15]. The shuttles themselves are passive executors, lacking any form of local intelligence.

The analysis revealed that SBSRS systems generally operate effectively under static and controlled conditions but experience significant challenges in more dynamic, large-scale and high-throughput environments. These challenges are particularly evident in congestion at shared resources, inflexibility in task allocation and coordination inefficiencies across agents and zones. Importantly, these limitations stem not merely from control policies but from the architectural design of SBSRS control systems.

At the same time, the rise of Industry 4.0 introduces a shift in warehouse design and control. I4.0 is characterized by technologies like CPS, IoT and AI, which enable real-time data sharing, distributed coordination and local decision-making [21, 7]. The system analysis showed that current SBSRS control architectures fail to align with these capabilities.

To guide this architectural transition, this study identifies five key improvement needs (IN1-IN5) for next-generation SBSRS, based on both literature and system-level analysis: (IN1) Distributed and autonomous decision-making; (IN2) Adaptive traffic and resource management; (IN3) Enhanced inter-agent coordination; (IN4) Modular and scalable system architecture; (IN5) Industry 4.0 readiness. These improvement needs serve as design requirements for evaluating and developing future control architectures.

This study uses Multi-Agent Systems (MAS) as a modeling framework to implement alternative control architectures that directly address the identified improvement needs. This agent framework provide a formal method for modeling systems composed of autonomous agents, entities that perceive their environment, make local decisions and interact with other agents or with centralized components [22]. Each agent operates independently yet contributes to system-level objectives through communication and coordination. In the context of SBSRS, MAS allow shuttles and other material handling units to act as intelligent agents capable of local routing, dynamic resource negotiation and local conflict resolution.

MAS is used in this study as a modeling paradigm to support the implementation of alternative control architectures for SBSRS. MAS allow the system to be represented as a collection of autonomous, interacting agents that can make local decisions and collaborate to achieve global objectives. This makes MAS particularly well suited to dynamic warehouse environments where modularity, responsiveness and distributed control are essential. Importantly, MAS is a modeling approach that offers the structural flexibility to implement a range of architectures by varying communication, authority and decision-making structures among agents. This makes it well-suited to design and compare SBSRS control strategies that address all five improvement needs.

B. Control Architectures in SBSRS

Control architecture refers to the distribution of decision-making authority and coordination mechanisms within a system. In the context of SBSRS, it determines how control responsibilities, including task assignment, routing and resource allocation, are distributed across system components. The structure of these interactions significantly impacts the system's performance. This section reviews the main control architectures that can be implemented using a MAS framework while integrating findings from both SBSRS-specific and related logistics literature.

The literature distinguishes four main MAS-based control architectures [11]:

1. Centralized Control Architectures

Centralized architectures rely on a single global controller, typically implemented as a Warehouse Control System (WCS),

to manage all decision-making in the system. Tasks are assigned top-down with all system components acting as passive executors. Most SBSRS studies follow this paradigm especially those that use heuristic or optimization-based models for shuttle routing and task sequencing.

For example, Carlo and Vis (2012) model a tier-captive SBSRS and demonstrate throughput improvements through heuristic scheduling of lifts, while Wang et al. (2015) propose a genetic algorithm for optimizing shuttle task assignment in multi-tier systems [2, 19]. Yang et al. (2015) integrates storage assignment with dynamic retrieval sequencing in a centralized control scheme [23]. These studies provide high-performance solutions for static or moderately dynamic systems but retain a strictly centralized control structure where shuttles remain passive executors.

2. Hierarchical Control Architectures

Hierarchical architectures introduce multiple layers of control (e.g., WMS → WCS → shuttles) each managing a specific subsystem or abstraction level. This decomposition improves scalability and task specialization but still relies on a top-down command flow. Shuttles and lifts typically remain reactive components, executing decisions made at higher levels.

Wang et al. (2016) and Cheng et al. (2023) implement hierarchical queuing and scheduling schemes for shuttle and lift operations, aligning with the architecture this study's use case ADAPTO [18, 13]. Although coordination occurs within subsystems (e.g., across all shuttles), individual components remain passive and lack autonomy. The architecture is structured but inflexible when facing dynamic conditions or shared-resource contention.

3. Distributed Control Architectures

Distributed control eliminates central authority and allows each agent (e.g., shuttle) to make local decisions based on its perception and peer interactions. This architecture provides high robustness and responsiveness, especially in dynamic environments. However, distributed systems may lack system-wide coherence if coordination mechanisms are not carefully designed.

In the SBSRS domain, fully distributed implementations are rare. Related studies in the broader logistics field offer some insights. For example, Turhanlar et al. (2022) present a distributed control framework for an aisle-changing AMR system that uses IoT-based inter-agent communication for collision and deadlock avoidance [4]. Although not SBSRS-specific, this model closely resembles the dynamics of tier-captive SBSRS operations and highlights the potential for local agent negotiation. However, the study does not address centralized oversight or system-wide performance metrics.

4. Hybrid Control Architectures

Hybrid architectures combine centralized and distributed decision-making. A high-level controller ensures system-level goal (e.g. throughput or fairness) while autonomous agents make local decisions for routing or conflict resolution. This layered approach aims to capture the advantages of both global coordination and local adaptability.

In logistics, hybrid architectures are increasingly applied but in SBSRS they remain relatively underexplored. Ekren et al. (2023) investigate a hybrid model for tier-to-tier, aisle-captive SBSRS, where one aisle-captive shuttle and one lift coordinate to synchronize tasks [3]. This is a meaningful architectural contribution but it addresses only a limited scope of agent autonomy and does not generalize to fully hybrid system-wide control. Another promising hybrid implementation is reported by Fan et al. (2025) who integrate real-time multi-agent path finding into a hierarchical control structure for a large-scale, grid-based SBSRS [5]. The study claims a 31.8% throughput improvement but the full methodology and results are not yet publicly available. Also the the system differs fundamentally from tier-captive SBSRS: it features 3D shuttle movement across a grid, rather than horizontal shuttle movement and vertical lift transport. The results are not directly transferable to the control challenges studied in this paper, however the study is conceptually relevant as evidence of MAS-based decision-making in storage systems.

Key Insights from Literature

A review of seventeen studies on SBSRS control architectures highlights four key observations. First, centralized and hierarchical architectures remain the dominant approach in both industrial systems and academic research. Second, these models consistently exhibit top-down control, passive shuttle behavior and global optimization as core traits. Third, although hybrid and distributed control architectures are increasingly explored in related domains such as AGVs and AMRs, their application to SBSRS remains limited. Fourth, and most critically, no comparative studies currently evaluate the performance of different control architectures in SBSRS, despite the architectural limitations identified in Section II.A.

TABLE I: COMPARISON OF CONTROL ARCHITECTURES FOR SBSRS AND OTHER MATERIAL HANDLING SYSTEMS

Architecture	Decision Level	Main Advantages	Main Limitations
Centralized	Global (WCS)	Global optimization, full system visibility	Limited scalability, bottlenecks, single-point failure risk
Hierarchical	Layered (WMS → WCS → Shuttle)	Structured control with some delegation	Still rigid, slow adaptation to dynamic changes
Distributed	Local (per shuttle or unit)	High modularity, responsiveness, fault tolerance	Weak global coordination, local conflicts possible
Hybrid	Mixed (local + global)	Combines flexibility and global goals; scalable	Complex integration, coordination tuning needed

Table I provides a comparative overview of these control architectures with its strengths and limitations in the context of SBSRS.

Rationale for Hybrid Control Architecture

Hybrid MAS control architectures emerge as a particularly promising solution for SBSRS, combining centralized oversight with distributed agent-level autonomy for localized responsiveness.

Hybrid architectures are well equipped to meet all five improvement needs (IN1-IN5) identified in Section II.A. They allow distributed and autonomous decision-making (IN1) by empowering shuttles to plan and execute routes independently. They support adaptive traffic and resource management (IN2) through real-time reactions to local congestion or task delays. Through agent-level communication and conflict resolution protocols, they fulfill the need for inter-agent coordination (IN3). Furthermore, hybrid MAS models support modular scalability (IN4), additional shuttles or layout extensions can be integrated without redesigning the entire control system. Finally, their compatibility with local data exchange and intelligent edge control aligns naturally with Industry 4.0 interoperability (IN5).

An additional benefit of hybrid MAS is its incremental deployability. Shuttle-level autonomy can be added to existing systems like Vanderlande's ADAPTO without changing the overall control architecture. This makes it possible to improve responsiveness and flexibility by enhancing only the shuttle behavior without redesigning the full system.

III. Methodology

This section presents the methodological framework used to design, test, evaluate and compare two control architectures for SBSRS: a hierarchical baseline and an I4.0-inspired hybrid model. It outlines the architectural concepts, conceptual designs, simulation implementation and experimental setup.

A. Conceptual Control Architectures and Operational Design

The two control architectures follow the same task lifecycle, task generation, task assignment, routing, item transfer and completion, they differ fundamentally in how decisions are distributed and executed. The conceptual design of the hybrid model is guided by five design objectives, derived from the SBSRS improvement needs. To clarify scope, this conceptual model focuses exclusively on shuttle operations. Other agents like vertical lifts and P&D-conveyors are abstracted to isolate the architectural effects on shuttle coordination.

In the hierarchical architecture all decision-making resides in a central controller, consistent with traditional Warehouse Control Systems. This controller continuously monitors system state, assigns tasks to idle shuttles, computes fixed paths using Dijkstra's algorithm and reserves all required nodes via resource objects. Cross-aisle access is regulated using a global fairness queue with "safe zones" to isolate blocked paths and avoid deadlock. Shuttles are passive units that follow commands without embedded intelligence.

This structure is visualized in Figure I, which illustrates the strict top-down control flow typical of hierarchical systems.

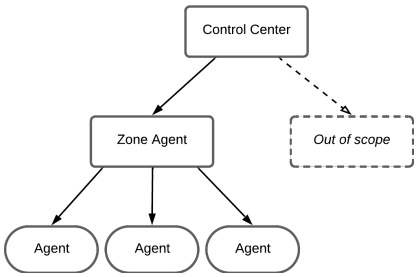


Figure I: Conceptual design of hierarchical control structure with higher-level coordination

In contrast, the hybrid architecture distributes control by equipping shuttles with local sensing, communication and decision-making capabilities. Each shuttle functions as an autonomous agent that evaluates incoming tasks using a utility function balancing distance, congestion and urgency. Tasks are selected through a peer-to-peer exchange protocol rather than top-down assignment. Agents compute routes dynamically based on real-time network state and congestion levels. For shared infrastructure such as cross-aisles, agents coordinate using message-passing and token-based fairness negotiation. For example, when multiple agents approach a shared aisle, they exchange state information and agree on entry order. If wait thresholds are exceeded agents reroute autonomously enabling localized deadlock avoidance and improved flow resilience.

This structure is depicted in Figure II showing agent-to-agent communication alongside a lightweight supervisory layer. The central controller maintains global task visibility and high-level throughput monitoring but no longer controls operational details.

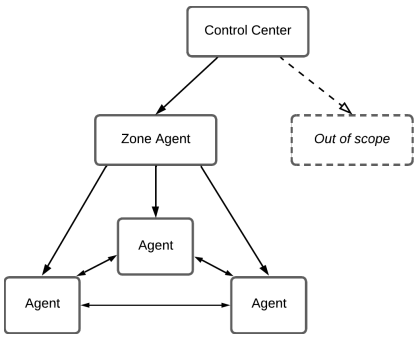


Figure II: Conceptual design hybrid control structure with agent autonomy and coordination

The operational control logic of both hierarchical and hybrid MAS architectures is defined through five core decision-making components which include task generation, task selection, routing, conflict handling and communication. The degree of control distribution and agent autonomy in each component affects the system's responsiveness and scalability together with its alignment to Industry 4.0 principles. Table II provides a high-level summary operational differences, including how decision-making responsibilities are distributed across the two architectures.

Table II: Comparison of Decision Logic in Hierarchical vs. Hybrid Control Architectures

Feature	Hierarchical Architecture	Hybrid Architecture
Shuttle Autonomy	None; shuttles passively execute top-down commands	Partial; shuttles operate as autonomous agents with local decision-making and perception
Communication	Top-down only (WCS to shuttle); no peer communication	Distributed; includes vertical communication (control center to agent) and peer-to-peer messaging
Decision-Making	**Centralized:** all logic handled by higher-level controller	**Hybrid (Mixed):** centralized task generation; distributed decision-making by shuttle agents
Task Generation	Centralized; WCS assigns tasks based on global system view	Centralized; same task source, but tasks are accessible to agents for selection
Task Assignment	Central controller assigns tasks using fixed rules (e.g., FCFS)	Distributed task selection based on agent utility functions (distance, load, wait time)
Routing	Fixed route pre-computed by controller; static path execution	Dynamic and locally adaptive routing based on live network state
Conflict Resolution	Static central rules; controller intervenes to resolve conflicts	Local negotiation protocols and rerouting strategies used by agents
Safety Constraint	One-shuttle-per-aisle enforced globally to prevent deadlocks	Agents maintain local safe distances; allow higher throughput through finer control

B. Modeling Approach

A hybrid simulation approach was employed, combining Discrete-Event Simulation (DES) and Agent-Based Modeling (ABM) to accurately represent the dynamics of SBSRS under different control architectures. This methodological choice was guided by a structured literature review and multi-criteria decision-making (MCDM) process, which assessed common modeling paradigms in SBSRS research against system characteristics such as event-driven operations, agent autonomy, stochastic task flows and dynamic routing behavior. The analysis confirmed that no single modeling method is sufficient to capture the full complexity of SBSRS; instead, a hybrid DES-ABM approach provides the most accurate and flexible representation.

Discrete-Event Simulation was used to model the system's operational flow, including queuing behavior, task generation, movement scheduling and node-level resource constraints. DES provides a structured, time-accurate foundation for modeling process-level events in SBSRS and is essential for evaluating system-level key performance indicators such as throughput and shuttle utilization.

Agent-Based Modeling complements DES by enabling the modeling of distributed behavior and inter-agent coordination. Each shuttle is implemented as an autonomous agent running an internal decision loop within the event-driven DES framework. This allows the agents to dynamically evaluate tasks, plan paths and resolve conflicts in real time based on local environmental information. ABM is especially critical for representing the hybrid MAS control architecture, as it captures emergent behavior arising from local interactions and supports distributed negotiation mechanisms required for congestion management and cross-aisle coordination.

This combination enables the simultaneous modeling of centralized logic (as in the hierarchical architecture) and distributed, autonomous behavior (as in the hybrid architecture) within a consistent environment. The approach offers the flexibility to compare both control strategies under identical physical conditions and operational scenarios.

C. Simulation Model and Implementation

The simulation model was developed in Python using SimPy for discrete-event process modeling and NetworkX for representing the system layout as a directed graph. The layout is based on a simplified, single-level instance of Vanderlande's ADAPTO system: several bidirectional storage aisles are perpendicularly connected to a shared front cross-aisle, which links to inbound and outbound lift interfaces as well as a designated safe zone. The safe zone functions as an isolated parking area where shuttles can temporarily wait or reroute in case of congestion or deadlock, without blocking the main operational network.

The simulation model is structured to separate general system elements from architecture-specific control logic. It captures a shared physical environment, a common shuttle behavior framework and centralized task generation, all of which remain consistent across experiments. While the physical system and agent framework remain fixed, architectural differences manifest in how and by who control functions are implemented, specifically task assignment, path planning, coordination and safety management. This modularity allows the general components to be used across experiments, ensuring consistent conditions when comparing the two control architectures.

The general system components can be grouped into four elements: the racking structure, resource constraints, task generation and shuttle behavior. The SBSRS layout, the racking structure, is encoded as a directed graph where nodes represent physical positions (storage slots, intersections or entry/exit points) and edges define valid shuttle movements. All edges are equally weighted to reflect fixed travel time between nodes. To prevent shuttle collisions each node is coded a resource of capacity one. The type of resource-locking differs by control ar-

chitecture: the hybrid MAS model uses fine-grained node-level locks, whereas the hierarchical model enforces a one-shuttle-per-aisle constraint through coarse-grained zone locking, consistent with the real-world ADAPTO system.

Tasks are centrally generated with a fixed 50/50 chance between storage and retrieval to maintain a steady state. In the hierarchical model they enter a central FIFO queue; in the hybrid MAS model they populate a shared pool (max. four per type). Both buffers are kept full to ensure uninterrupted operation. All agents follow a common behavioral state-chart comprising the following sequence: idle → task selection → routing and movement (incl. waiting and deadlock resolution) → item handling → completion → idle. This forms the foundation for both control architectures, with architecture-specific logic injected at the task selection, routing and coordination stages.

In the hierarchical architecture, a central controller orchestrates the full task life cycle. It monitors idle shuttles and a FIFO task queue, assigns tasks based on global distance heuristics and computes full routes using Dijkstra's algorithm. All nodes along the selected path are locked in advance through pessimistic reservation to ensure collision-free execution. The controller manages cross-aisle access through a fairness mechanism and proactively monitors for deadlock by detecting immobile shuttles. If a shuttle is blocked on the cross-aisle for longer than a set threshold it is rerouted to the safe zone. Communication is strictly unidirectional with the controller issuing commands to passive shuttle agents that execute them without local adaptation.

In contrast, the hybrid MAS architecture distributes decision-making across autonomous shuttle agents. Tasks are selected by idle agents from a shared task pool using a utility-based evaluation function:

$$U_{ij} = -\left(\alpha \cdot d_{ij}^{\text{route}} + \beta \cdot c_{ij}^{\text{predicted}} + \gamma \cdot w_j\right) \tag{1}$$

where d_{ij}^{route} is the shortest distance from shuttle i to the start location of task j, $c_{ij}^{\text{predicted}}$ is the estimated congestion along the route and w_j is the current waiting time of task j. The weights α, β and γ balance the relative influence of distance, congestion and task age in the selection process. Agents compute utility scores in parallel and atomically select the task with the highest utility ensuring local load balancing and task differentiation. Path planning is local and adaptive. Agents forecast their intended paths and write these predictions into a shared reservation map. Conflicts are resolved dynamically: shuttles maintain a rolling two-node reservation window and yield access based on fairness policies such as proximity to the cross-aisle. If conflicts persist or predicted overlaps exceed a defined threshold agents initiate rerouting protocols. This distributed coordination mechanism supports responsive, congestion-aware routing without the need for centralized intervention.

All agent logic, including task evaluation, conflict resolution and re-routing, is executed as SimPy generator processes enabling concurrent and asynchronous decision-making within the discrete-event simulation framework. Communication among agents occurs via lightweight message-passing and shared data structures which preserves simulation efficiency while enabling real-time coordination.

To isolate the effect of architectural design, the model assumes ideal operating conditions with no equipment failures, communication delays or stochastic disturbances. Only a single-level, tier-captive SBSRS is simulated, excluding lift coordination and buffering. Tasks are continuously available, shuttles have homogeneous capabilities and movement occurs at constant speed with fixed handling times. The system operates in steady-state with a balanced 50/50 mix of storage and retrieval tasks. These simplifications provide a scientifically valid environment for architecture-level comparison while acknowledging that the model does not aim to replicate the full complexity of real-world ADAPTO systems.

D. Simulation Setup and Verification

The experimental setup for simulation followed a rigorous and statistically grounded design to enable a controlled comparison between the hierarchical and hybrid MAS architectures. Parameter choices were formed by established methods in simulation modeling. A warm-up period of 1,200 simulation seconds was determined using Welch's method to remove initialization bias, followed by a run length of 12,000 seconds per replication based on the rule-of-thumb of 10× warm-up duration [20, 9]. 40 replications were conducted for each scenario, selected through convergence testing and the doubling rule to ensure statistical confidence.

Experimental scenarios varied along two dimensions: system size and shuttle density. Increasing the system size (`Aisles x Slots`), including 10×10, 15×15, 20×20 and 25x25 grids, allowed assessment of routing complexity and architectural scalability. Shuttle density was varied by simulating 2 to 6 shuttles operating concurrently within the system, testing the system's ability to handle congestion and coordinate shared-space access, particularly within the cross-aisle.

The key performance indicator (KPI) is system throughput expressed as completed tasks per simulation hour. Additional performance indicators included operational efficiency (throughput per distance traveled), scalability (sensitivity to layout and shuttle count) and robustness (variance in throughput across replications). These KPIs align with performance assessment criteria commonly used in automated logistics systems literature.

Model correctness was verified through unit testing of shuttle behavior, deadlock recovery and path execution. Scenario-based testing with 1-2 shuttles ensured throughput aligned with expected theoretical behavior. Traces were manually inspected to confirm correct locking, routing and task completion. Although real-world data from the use case is available, empirical validation was not pursued as direct comparison would not be meaningful. This study isolates and simplifies shuttle-level logic in a single-tier setup, whereas the actual ADAPTO system includes vertical lifts, complex scheduling and full-system control integration. Nonetheless, the hierarchical control logic implemented in this study reflects the principles used in ADAPTO's shuttle operations and the hybrid control model is based on both agent-based literature and system-specific constraints. Since the objective is to compare control architectures rather than replicate exact performance, model verification and consistency with known use case control behaviors were considered sufficient.

IV. Results and Discussion

This section presents and interprets the results of the simulation comparing the performance of hierarchical and hybrid control architectures.

A. Throughput

System throughput is the key indicator of SBSRS capacity. Figure III shows how hybrid control improves throughput over hierarchical control, reported as both percentages and absolute values. Across all scenarios hybrid outperforms hierarchical. In a small 10×10 system with two shuttles it completes about 24 more tasks per hour (+15%), while in a 25×25 system with six shuttles the gain exceeds 52 tasks per hour (+90%).

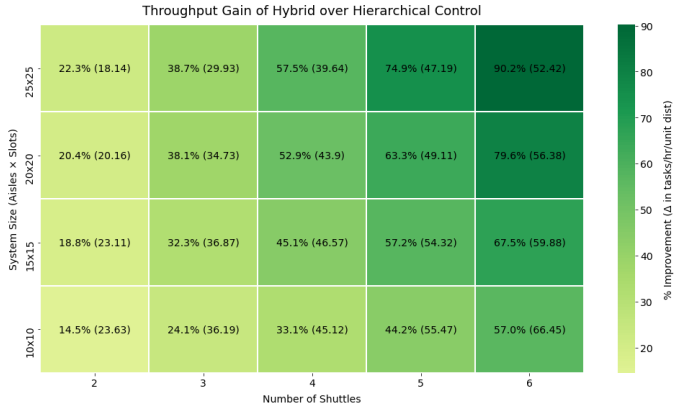


Figure III: Throughput Improvement of Hybrid over Hierarchical Architecture [% and Tasks/hour]

Two clear trends emerge. First, relative gains grow with system size, as centralized coordination in hierarchical control increasingly becomes a bottleneck. Second, the gap widens further with higher shuttle density: while hierarchical control suffers from congestion and queuing, hybrid control leverages distributed decision-making to keep shuttles moving in parallel. Overall, hybrid architectures not only achieve higher throughput but also scale better with system size and complexity.

B. Scalability

The heatmap in the previous section already showed that hybrid architectures achieve consistently higher throughput than hierarchical control, especially as system size and shuttle density increase. To better understand why this happens, the scalability analysis examines how throughput changes with higher shuttle density in a fixed layout and with growing system size at a fixed fleet size.

Scalability is first examined by increasing shuttle density in a fixed 20×20 layout (Figure IV). Under hierarchical control throughput peaks at 98 tasks/hour with two shuttles but then declines steadily, reaching only 70 tasks/hour with six shuttles. This drop is caused by congestion and path conflicts that the centralized controller cannot resolve effectively. Hybrid control behaves very differently. Throughput keeps rising up to 128 tasks/hour with five shuttles and only levels off slightly at six (126 tasks/hour). The performance gap therefore widens with density from about +20% at two shuttles to nearly +80% at six, showing how distributed routing and local negotiation enable better congestion handling and parallel execution.

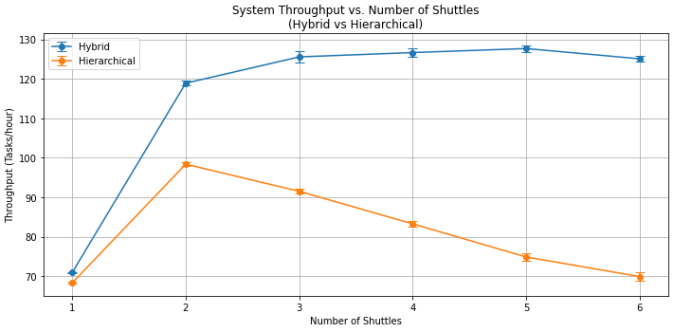


Figure IV: System Throughput over Number of Shuttles. Fixed Layout 20×20. Hybrid and Hierarchical architecture.

When scaling system size with a fixed amount of shuttles both architectures follow the same trend. They lose throughput due to longer travel distances but hybrid consistently maintains higher performance. This confirms that while absolute capacity declines in larger layouts hybrid architectures sustain a clear advantage across scales.

These results show that hierarchical control cannot scale effectively with density or size, while hybrid architectures sustain high throughput by reducing congestion and coordinating resources more efficiently. This confirms hybrid's suitability for large, high-density SBSRS environments.

C. Operational Efficiency

Raw throughput measures how many tasks are completed but it does not reflect the travel effort required. To make a fair comparison across different system sizes and shuttle configurations, normalized throughput is used, defined as tasks per hour per unit shuttle travel distance. This metric shows how effectively each architecture converts movement into productive work.

Figure V presents the efficiency gain of hybrid over hierarchical control. Hybrid consistently delivers higher normalized throughput across all cases with the advantage growing as systems become larger and denser. In small layouts such as a 10×10 with two shuttles, efficiency improves by about 16%, while in a 25×25 system with six shuttles the improvement reaches 68%.

The analysis reveals three clear patterns. Efficiency gains increase with shuttle count as hybrid avoids the traffic jams that arise under centralized logic. They also grow with system size: although absolute efficiency declines with longer travel distances, hybrid reduces wasted movement much more effectively. Finally, while efficiency always drops in larger layouts, hybrid sustains higher productivity levels, confirming its superior use of shuttle capacity in demanding conditions.

Overall, hybrid MAS does not just raise output it also makes shuttle movements far more productive.

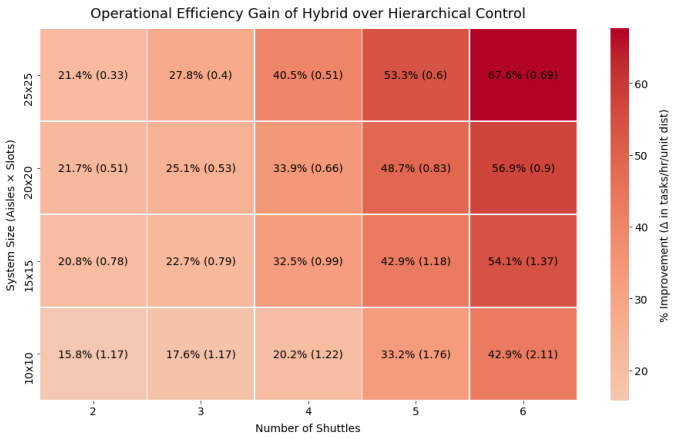


FIGURE V: HEATMAP OF NORMALIZED THROUGHPUT IMPROVEMENT OF HYBRID MAS CONTROL OVER HIERARCHICAL CONTROL (% AND ABSOLUTE GAIN IN TASKS/HOUR/UNIT DISTANCE), ACROSS SYSTEM SIZES AND SHUTTLE COUNTS.

D. Best Achievable Throughput and Efficiency

The heatmaps in previous sections compared both architectures under identical conditions but each may perform best at a different shuttle count. To address this, Figure VI report the best achievable throughput for each system size based on the optimal shuttle count of each architecture.

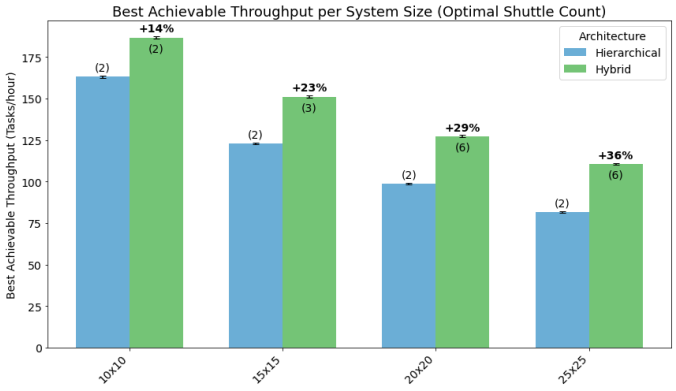


FIGURE VI: BEST ACHIEVABLE THROUGHPUT PER SYSTEM SIZE, USING THE OPTIMAL SHUTTLE COUNT PER ARCHITECTURE.

Results show that hybrid control consistently outperforms hierarchical control at its best operating point. Throughput gains rise from +14% in 10×10 systems to +36% in 25×25 layouts, while best normalized efficiency improves by 16-22% across all sizes. In several cases, hybrid reaches its optimum at higher shuttle counts, confirming its ability to coordinate larger fleets without performance loss.

Overall, these findings highlight that hybrid MAS not only achieves higher peak performance but also uses additional resources more effectively, reinforcing its suitability for large and dense SBSRS environments.

E. Robustness

Robustness refers to how stable and predictable performance remains under repeated conditions. This is critical in industrial settings where warehouses rely on consistent throughput to ensure reliable operations. Robustness was assessed over 40 replications per scenario using the interquartile range (IQR) of throughput and the coefficient of variation (CV).

Both architectures maintain narrow IQRs of about 6-8 tasks/hour, but hybrid consistently produces tighter ranges, shorter whiskers and no outliers. By contrast, the hierarchical model shows both low outliers (linked to congestion or near-deadlock) and occasional high outliers (favorable task sequences), reflecting greater sensitivity to stochastic variation.

The CV results emphasize this difference. For hybrid, values remain consistently low (2.2-2.8%), while hierarchical control starts at 2.7% with two shuttles and rises steadily to 6.0% at six shuttles. Although these percentage shifts may seem small, simulation literature highlights that even one or two percentage points in CV can mark substantial differences in robustness particularly in tightly coupled systems where variability compounds over time. Reporting CV instead of standard deviation also allows a fairer comparison by normalizing variability to mean performance.

Overall, hybrid MAS delivers not only higher throughput but also more predictable performance which is equally vital for real-world SBSRS operations.

F. Summary of Key Findings

Table III summarizes the simulation results across the four core performance metrics. It compares the hierarchical and hybrid MAS control architectures across all tested scenarios and highlights the relative advantages of each.

G. Comparative Interpretation and Discussion

The simulation experiments clearly show that the choice of control architecture has a major impact on SBSRS performance. Across all scenarios the hybrid MAS architecture consistently outperforms the hierarchical baseline although the reasons for these gains differ across the various KPIs.

For throughput, hierarchical control peaks at low shuttle counts and quickly levels off. This comes from its centralized scheduling approach where global path reservations and sequential task assignment create bottlenecks once multiple shuttles compete for resources. After a certain point adding more shuttles no longer increases capacity since congestion and blocking dominate system behavior. In contrast, hybrid control continues to benefit from additional shuttles. Local routing and distributed task allocation allow shuttles to operate in parallel, which explains why throughput keeps improving up to five or six shuttles before stabilizing [10].

When scaling system size both architectures show reduced performance due to longer travel distances but the drop is steeper in the hierarchical model. Centralized coordination struggles with the growing complexity of larger networks where global planning becomes rigid and prone to bottlenecks. Hybrid control sustains higher throughput at all sizes even though absolute performance still decreases in bigger systems. This

Table III: Quantitative and Qualitative Summary of Results Across Control Architectures

Aspect	Hierarchical Architecture	Hybrid Architecture	Net Effect
Throughput	Peaks early then declines: 98 → 70 tasks/hour from 2 to 6 shuttles due to congestion.	Scales effectively with more shuttles: peaks at 128 tasks/hour (5 shuttles), remains high at 6.	Hybrid achieves 30-60% higher throughput in large layouts; up to +52 tasks/hour (+90%) in extreme cases.
Scalability	Throughput halves from 152 to 77 tasks/hour when scaling from 10×10 to 25×25 (3 shuttles); limited by coordination bottlenecks.	Throughput declines more gradually: 186 to 109 tasks/hour; performs well even in large systems.	Hybrid scales reliably without saturation; avoids bottlenecks seen in centralized control.
Operational Efficiency	Normalized throughput drops from 8.1 to 1.1 tasks/hour/unit distance as congestion increases; efficiency loss due to idle travel.	Declines from 8.6 to 1.7 tasks/hour/unit distance, but more gradually; efficient routing reduces waste.	Hybrid maintains up to +68% higher movement efficiency in dense scenarios; less wasted motion.
Robustness	IQR: 7.4-8.0 tasks/hour; CV: 6.9-7.2%; 1-2 outliers per scenario with 3+ shuttles, some < 90 tasks/hour.	IQR: 6.8-7.1 tasks/hour; CV: 5.6-5.7%; no outliers across all cases.	Similar central variability, but hybrid eliminates low-performance outliers and reduces CV by 1.5%.
Behavioral Insights	Centralized FIFO leads to sequential aisle use, blocking and frequent deadlocks with 3+ shuttles.	Agents negotiate path access and reroute dynamically, dispersing congestion before deadlocks emerge.	Hybrid enables adaptive, scalable coordination aligned with Industry 4.0 principles; hierarchical control becomes rigid under load.

means hybrid scalability comes not from avoiding performance loss completely but from keeping throughput at a consistently higher level as warehouses grow [16].

Operational efficiency highlights another difference. Hierarchical control suffers from wasted travel and idle time as shuttles often wait for entire path reservations to clear. This inefficiency grows in larger and denser systems where more shuttles compete for limited cross-aisle capacity. Hybrid control reduces this wasted effort by letting shuttles adapt their routes to local conditions and share workload more effectively. As a result shuttle movements translate more directly into productive work explaining the clear efficiency advantage seen in the normalized throughput results [16].

Robustness further illustrates the stability of hybrid performance. While both models show similar central spreads, the hierarchical system often produces outliers including runs with much lower throughput due to congestion spirals or poor task–shuttle matches. Hybrid control avoids these low-performing cases and keeps variability lower overall leading to more predictable system output. In practice, this reliability is as important as high average performance since industrial operations depend on stability and worst-case guarantees.

Two broader observations also stand out. First, in small and simple systems with only a few shuttles, the performance differences between hierarchical and hybrid control remain limited. This suggests that centralized control may still be suitable in low-demand settings while the benefits of hybrid control become more important as systems scale in size and complexity. Second, the higher movement efficiency of hybrid control points to possible long-term benefits such as lower energy consumption and reduced mechanical wear. These factors were not modeled directly but they indicate advantages that go beyond throughput alone.

Overall, the results show that the performance gap between hierarchical and hybrid control is not caused by isolated factors but by fundamental architectural differences. Hierarchical control is limited by its centralized nature which restricts scalability, increases inefficiency and exposes the system to variability under load. Hybrid MAS, by contrast, uses distributed autonomy and local coordination to maintain throughput, efficiency and stability across a wide range of conditions. This makes hybrid architectures better suited to the dynamic and high-throughput demands of modern SBSRS and aligns them closely with the goals of I4.0.

While the performance advantages of hybrid MAS are clear across all tested scenarios, the extent to which these results can be generalized must be considered carefully. The results apply most directly to SBSRS with layouts, task patterns and operating rules similar to those modeled here. Still, the architectural mechanisms driving the differences between hybrid and hierarchical control like congestion management, distributed decision-making and robustness are not unique to the simulated system. These are structural properties of control architectures that would influence performance in a wide range of SBSRS configurations.

This means that even though the exact throughput values may vary with different warehouse geometries, demand profiles or multi-level system designs, the relative trends are likely to remain the same. Hierarchical control struggles with scalability and variability while hybrid maintains higher throughput, efficiency and stability. The generalizability of the findings is therefore rooted less in the specific case parameters and more in the demonstration of how architectural design fundamentally shapes performance dynamics across scales. In that sense, this study provides insights that extend beyond the particular system modeled offering lessons applicable to other SBSRS layouts and, more broadly, to automated material-handling systems that rely on fleets of cooperating vehicles.

V. Conclusion and Future Work

This study evaluated the impact of alternative control architectures on the operational performance of Shuttle-Based Storage and Retrieval Systems with a particular focus on Industry 4.0-oriented designs. Through a comparative simulation of a conventional centralized hierarchical control architecture and a I4.0-inspired hybrid control architecture, the work aimed to generate scientifically grounded insights on how architectural structure impacts operational system performance by assessing throughput, efficiency, scalability and robustness.

The simulation results demonstrate that control architecture selection strongly affects SBSRS performance. The hybrid model consistently outperforms the hierarchical architecture across all performance metrics and scenarios. The performance gap was most pronounced in large-scale environments operating with a high number of concurrent shuttles. The main findings can be summarized as follows:

- **Throughput:** Up to 90% higher in large-scale, high-density systems; 30-60% higher in mid-sized configurations; up to 36% higher under optimal comparable conditions.
- **Efficiency:** Up to 68% improvement in normalized throughput (tasks per distance unit).
- **Scalability:** Sustained or improving performance as system size and shuttle count increased, unlike hierarchical control which often degraded beyond two or three shuttles.
- **Robustness:** Significantly lower variance across replications, with hybrid control eliminating the low-outlier runs seen under centralized coordination.

In conclusion, this research shows that the way control is structured in an SBSRS has a significant impact on its operational performance. Using an ADAPTO-inspired case study, the results demonstrate that the hybrid model is more scalable, more efficient and more robust. The performance gap is most evident in large and highly dynamic systems where hierarchical control struggled with congestion and coordination demands. Hybrid MAS control provides a viable and future-proof alternative that not only improves performance but also supports the integration of Industry 4.0 technologies such as local autonomy, intelligent routing and real-time communication.

The key takeaway from this work is therefore the following: the control architecture itself is a critical design decision in SBSRS and hybrid control architectures offer a clear technical advantage and a strong basis for future I4.0 integration.

Despite these promising results, the study is subject to several limitations. The simulation focuses solely on shuttle agents in a single-level, tier-captive layout, excluding vertical lifts, buffering zones and multi-level routing layers. The use case reflects a specific SBSRS configuration (tier-captive, single cross-aisle) and other system variants were not considered. Idealized operating conditions were assumed, excluding disturbances such as equipment failures, communication delays and task cancellations, to isolate architectural effects. No cost-benefit analysis was conducted; although hybrid MAS offers technical advantages, its implementation may involve additional hardware, integration, or software costs that require economic justification. The control logic was abstracted: while the hierarchical model reflects simplified industrial practices (ADAPTO) and the hybrid MAS is based on literature, both exclude low-level operational detail to preserve comparability and scientific clarity. Finally, no empirical validation was performed, as the available data was not suitable for benchmarking architectural effects within the study's limited scope.

From a scientific perspective, it addresses two notable research gaps: first, the lack of comparative studies evaluating different control architectures in SBSRS; and second, the limited integration of Industry 4.0 concepts, such as local autonomy and agent-based coordination, within SBSRS research [11, 3]. While these topics have been explored qualitatively or in related logistics domains, systematic, quantitative comparison within SBSRS has remained limited.

Future research should extend the current model to multi-level SBSRS configurations that include vertical lifts, buffering zones and complex routing layers, which are expected to further amplify the benefits of hybrid control according to literature since they are proven effective in coordinating heterogeneous agents. Introducing stochastic disturbances such as equipment failures or communication noise would allow for a more realistic evaluation of system robustness, where hybrid control is likely to excel according to literature thanks to its distributed responsiveness and local fault tolerance. Further refinement of agent coordination mechanisms, including adaptive utility functions, communication strategies and learning algorithms, could enhance responsiveness and autonomy. Also, empirical validation through detailed system-level simulation or phased pilot implementations in real-world settings would help translate the conceptual findings into practice. Additionally, future work could explore the hardware and infrastructure requirements for the shuttle agent, such as onboard sensing, IoT connectivity and real-time localization, needed to enable the I4.0 capabilities assumed in this study. Lastly, in addition to technical performance, cost-benefit analyses covering CAPEX, OPEX and ROI are needed to assess the practical viability of hybrid MAS architectures.

Taken together, this work offers a first step toward rethinking control architectures for SBSRS in line with the adaptive, distributed and intelligent systems envisioned by I4.0.

References

[1] Kaveh Azadeh, René De Koster, and Debjit Roy. "Robotized and automated warehouse systems: Review and recent developments". In: *Transportation Science* 53.4 (2019), pp. 917–945.

[2] Héctor J Carlo and Iris FA Vis. "Sequencing dynamic storage systems with multiple lifts and shuttles". In: *International Journal of Production Economics* 140.2 (2012), pp. 844–853.

[3] Banu Y Ekren. "Transaction Processing Policies in a Flexible Shuttle-based Storage and Retrieval System by Real-time Data Tracking under Agent-based Modelling". In: (2023).

[4] Banu Y Ekren, Berk Kaya, and Melis Küçükyaşar. "Shuttle-based storage and retrieval systems designs from multi-objective perspectives: total investment cost, throughput rate and sustainability". In: *Sustainability* 15.1 (2022), p. 762.

[5] Hongkai Fan et al. “A Lifelong Multi-Shuttle Scheduling Framework for the AS/RS System”. In: *IEEE Transactions on Automation Science and Engineering* (2025).

[6] Maarten van Geest, Bedir Tekinerdogan, and Cagatay Catal. “Design of a reference architecture for developing smart warehouses in industry 4.0”. In: *Computers in industry* 124 (2021), p. 103343.

[7] Petri Helo and Yuqiuge Hao. “Artificial intelligence in operations management and supply chain management: An exploratory case study”. In: *Production planning & control* 33.16 (2022), pp. 1573–1590.

[8] René de Koster. “Warehousing 2030”. In: *Global Logistics and Supply Chain Strategies for the 2020s: Vital Skills for the Next Generation* (2022), pp. 243–260.

[9] Averill M Law. *Simulation modeling and analysis*. Vol. 5. Mcgraw-hill Education, 2015.

[10] Paulo Leitão. “Agent-based distributed manufacturing control: A state-of-the-art survey”. In: *Engineering applications of artificial intelligence* 22.7 (2009), pp. 979–991.

[11] Paulo Leitão and Stamatis Karnouskos. “Industrial agents: emerging applications of software agents in industry”. In: (2015).

[12] Yi Li and Zhiyang Li. “Shuttle-Based Storage and Retrieval System: A Literature Review”. In: *Sustainability* 14.21 (2022), p. 14347.

[13] Jia Mao et al. “Modelling analysis of a four-way shuttle-based storage and retrieval system on the basis of operation strategy”. In: *Applied Sciences* 13.5 (2023), p. 3306.

[14] László Monostori et al. “Cyber-physical systems in manufacturing”. In: *Cirp Annals* 65.2 (2016), pp. 621–641.

[15] Sascha Julian Oks et al. “Cyber-physical systems in the context of industry 4.0: A review, categorization and outlook”. In: *Information Systems Frontiers* 26.5 (2024), pp. 1731–1772.

[16] Adrià Salvador Palau, Maharshi Harshadbhai Dhada, and Ajith Kumar Parlikad. “Multi-agent system architectures for collaborative prognostics”. In: *Journal of Intelligent Manufacturing* 30.8 (2019), pp. 2999–3013.

[17] Lihle N Tikwayo and Tebello ND Mathaba. “Applications of industry 4.0 technologies in warehouse management: A systematic literature review”. In: *Logistics* 7.2 (2023), p. 24.

[18] Yanyan Wang, Shandong Mou, and Yaohua Wu. “Storage assignment optimization in a multi-tier shuttle warehousing system”. In: *Chinese journal of mechanical engineering* 29.2 (2016), pp. 421–429.

[19] Yanyan Wang, Shandong Mou, and Yaohua Wu. “Task scheduling for multi-tier shuttle warehousing systems”. In: *International Journal of Production Research* 53.19 (2015), pp. 5884–5895.

[20] Peter D Welch. “The statistical analysis of simulation results”. In: *Computer performance modeling handbook* (1983).

[21] Sven Winkelhaus and Eric H Grosse. “Logistics 4.0: a systematic review towards a new logistics system”. In: *International journal of production research* 58.1 (2020), pp. 18–43.

[22] Michael Wooldridge. *An introduction to multiagent systems*. John wiley & sons, 2009.

[23] Peng Yang et al. “An integrated optimization of location assignment and storage/retrieval scheduling in multi-shuttle automated storage/retrieval systems”. In: *Journal of Intelligent Manufacturing* 26 (2015), pp. 1145–1159.

B

Alignment of ADAPTO Challenges with SBSRS Improvement Needs

Confidential.

C

Overview of Modeling Approaches

Appendix C summarizes different modeling approaches considered for SBSRS and their strengths and limitations.

Method	Application Scope	Strengths	Limitations
Discrete Event Simulation (DES)	Processes with distinct sequential events. (Event-driven)	Captures high-level operational flow, effective for analyzing 'what-if' scenarios and throughput evaluation. Well-suited for process flows and queuing systems.	Struggles with modeling distributed decision-making and adaptive behaviors. Lacks autonomy for decentralized agents.
Agent-Based Modeling (ABM)	Complex interactions between autonomous agents. (Agent-centric)	Suitable for distributed control, emergent behaviors, and real-time adaptability.	Computationally intensive for large scale systems.
System Dynamics (SD)	Long-term resource management, policy analysis. (Feedback-driven)	Effective for modeling high-level feedback loops and resource flow.	Lacks granularity needed for detailed component interactions, not suited for real-time adaptability.
Petri Nets	Concurrent process modeling, workflow analysis. (Graphical, Mathematical)	Good for modeling concurrent processes, synchronization, and detecting deadlocks.	Limited in representing complex decision-making and decentralized, adaptive behaviors.
Queuing Network	Analytical modeling of waiting lines and service processes. (Stochastic/Mathematical)	Mathematically tractable, useful for performance analysis of queuing systems. Widely used in networked systems and operations research.	Assumes simplified behavior; limited ability to model dynamic control or adaptive agent behavior.
Hybrid Modeling (DES + ABM)	Systems requiring both high-level and agent-level interactions. (Combined Top-down + Bottom-up)	Combines high-level flow (DES) with adaptive, agent-level interactions (ABM). Balances process-driven modeling with agent autonomy.	Requires careful synchronization between DES and ABM components.
Digital Twins	Real-time monitoring, predictive maintenance, system optimization. (Real-time Virtual Representation)	Provides real-time system representation, supports predictive maintenance and operational optimization.	Resource-intensive, challenging to accurately replicate the real system.

Table C.1: Overview of Modeling Methods for SBSRS with their Strengths and Limitations

D

Simulation

Appendix D presents a couple of algorithms used in the simulation model to implement shuttle resource allocation, deadlock resolution and throughput computation. Next to that, the state charts and class diagrams of the simulation model are shown. These diagrams clarify control flow, locking, and data exchange in the simulation.

D.1. Pseudo Codes

Algorithm 2: Shuttle Resource Allocation for Controlled Movement

This algorithm ensures safe and congestion-free shuttle movement through the network. It controls access to nodes, enforces one-shuttle occupancy, and simulates transfer delays at cross-aisles.

Algorithm 2 Shuttle Resource Allocation for Controlled Movement

Input: SBSRS network, shuttle's current position, target position
Output: Controlled and congestion-free shuttle movement

```
1:  Initialize current_position                                   // Shuttle's starting location
2:  Compute shortest_path from current_position to target_position   // Path planning
3:  for each node in shortest_path do                             // Loop through all nodes in path
4:      Request access to node                                    // Ensure exclusive access before moving
5:      if node is occupied then                                  // Check if another shuttle is occupying the node
6:          Wait until node becomes available                     // Enter waiting state if blocked
7:      end if
8:      Move shuttle to node                                      // Update shuttle position
9:      Release access to previous node                           // Free up node for other shuttles
10:     if node is a cross-aisle transition then                  // Special condition for aisle switching
11:         Apply fixed transfer delay                            // Simulates wheel reconfiguration
12:     end if
13: end for
14: Return final shuttle position
```

Algorithm 3: Deadlock Resolution via Safe Zone

This algorithm detects deadlock situations in cross-aisles and resolves them by rerouting shuttles to a designated safe zone. It guarantees that blocked shuttles can resume their tasks without system-wide halts.

Algorithm 3 Deadlock Resolution via Safe Zone

Input: Shuttle states, simulation $current\_time\ t$, timeout threshold Δt
Output: Conflict resolution by rerouting shuttle to safe zone

while simulation is running **do**
  for all shuttles s in system **do**
    if s is in CrossAisle AND not $s.is\_resolving\_deadlock$ **then**
      if $current\_time - s.last\_moved\_time > \Delta t$ **then**
        Mark $s.is\_resolving\_deadlock \leftarrow$ True
        Save s's current task $\rightarrow s.saved\_task$
        Cancel active task and mark $s.busy \leftarrow$ False
        Compute path from $s.current\_position$ to CA-Safe
        for all nodes n in path **do**
          Wait for resource at n to become available
          Move s to n
        end for
        Set $s.in\_safe\_zone \leftarrow$ True
        Release any held zone or resource
      end if
    end if
  end for
  wait 1 time unit
end while

Algorithm 4: Compute System Throughput

This algorithm calculates throughput in tasks per hour based on the total number of completed tasks and simulation runtime. It provides the main performance metric for comparing control architectures.

Algorithm 4 Compute System Throughput

Input: Simulation time, total tasks completed
Output: Throughput Tasks per Hour ($Tasks/h$)

Initialize $total\_tasks\_completed = 0$ // Initialize counter
for each completed task **do** // Loop through all completed tasks
  Increment $total\_tasks\_completed$ by 1 (+= 1)
end for
Compute $throughput = total\_tasks\_completed/(simulation\_time/3600)$
Return $throughput$

D.2. Simulation Diagrams

This appendix summarizes the behavior and structure of the two control architectures used in the study. Figures D.1 and D.2 are state charts that capture per-shuttle logic (operational phases, guards in brackets, and fault sinks). Figures D.3 and D.4 are class diagrams showing the main modules, ownership/dependencies, and where KPIs and locks live. Notation: rounded boxes = states/classes; grey labels on arrows = events/guards; dashed links = non-owning dependencies or data access.

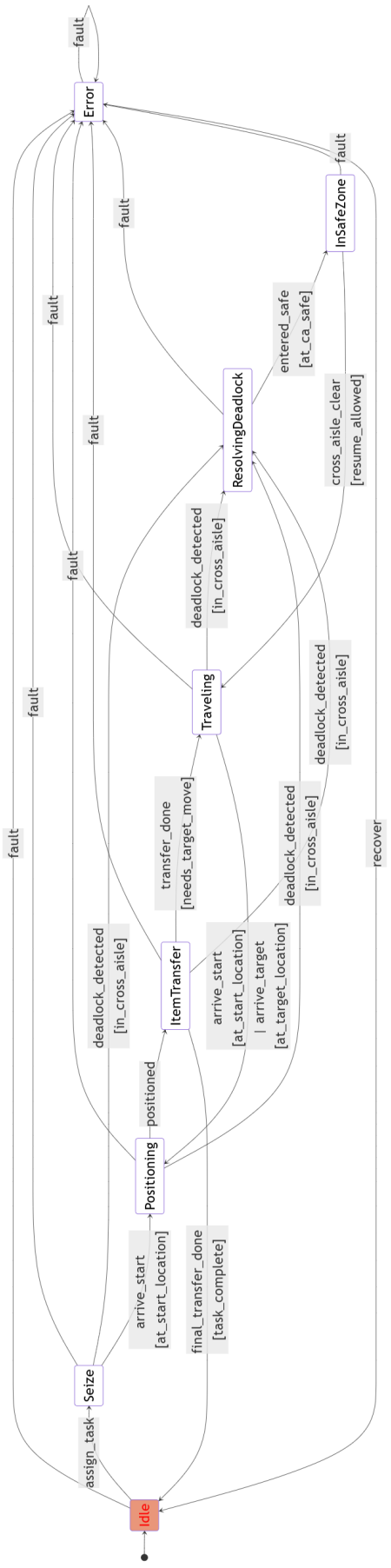


Figure D.1: State Chart of Hierarchical Architecture Simulation. With a central dispatcher, a shuttle receives assign_task and executes Seize → Positioning → ItemTransfer → Traveling, repeating for start/target legs. Cross-aisle deadlocks trigger ResolvingDeadlock and relocation to InSafeZone; recovery on cross_aisle_clear resumes the task. All unexpected conditions route to Error.

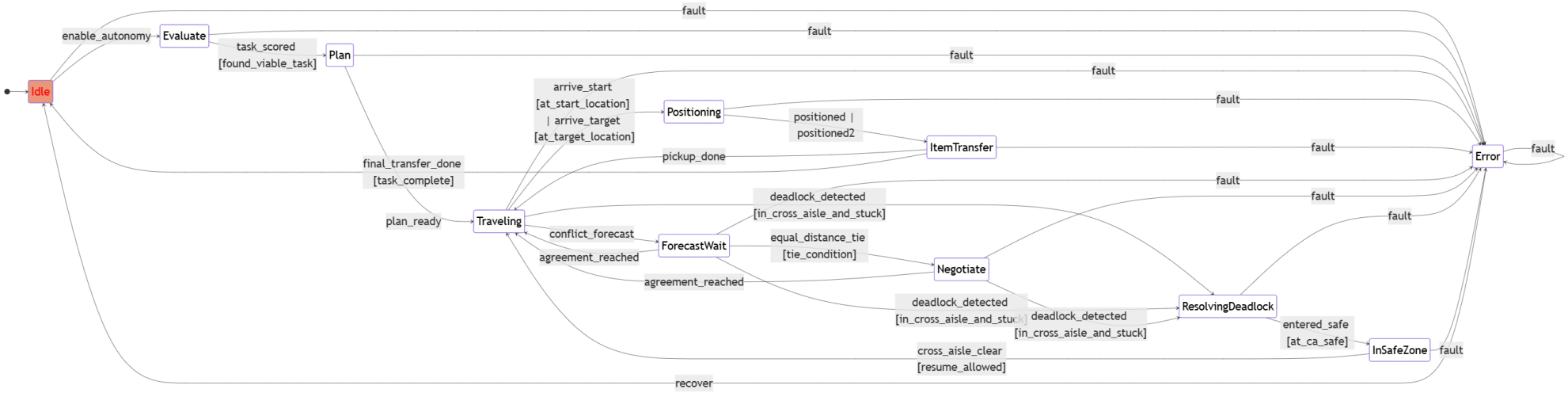


Figure D.2: State Chart of Hybrid Architecture Simulation. A shuttle autonomously evaluates and plans a task, forecasts its path, and (if needed) negotiates to avoid conflicts. Normal flow is Idle → Evaluate/Plan → Traveling → Positioning → ItemTransfer, looping until task_complete. Guards include arrive_start/arrive_target, conflict_forecast, tie_condition, and in_cross_aisle_and_stuck. Deadlock handling moves the agent to ResolvingDeadlock and InSafeZone until resume_allowed. Any fault transitions to Error.

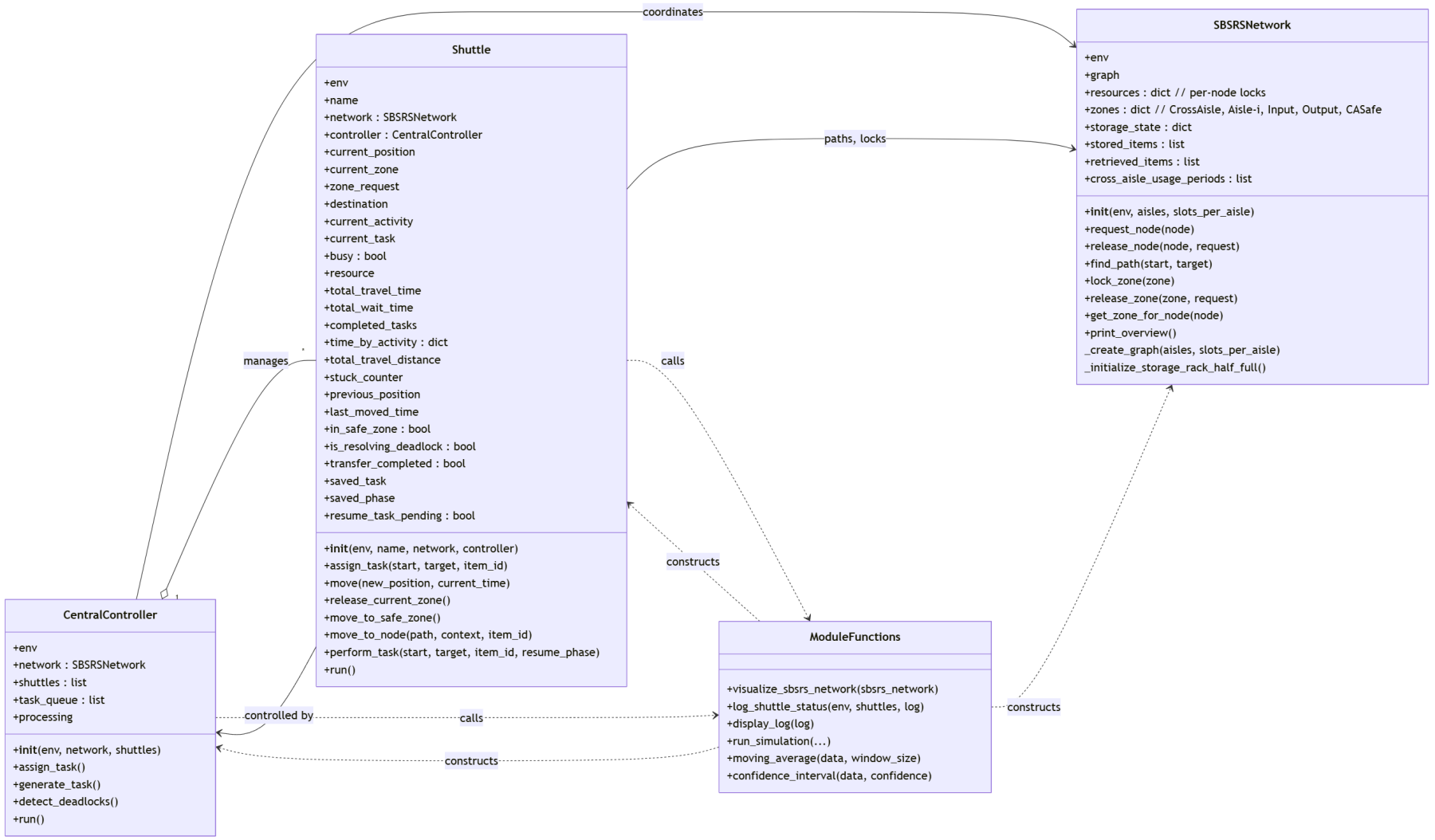


Figure D.3: Class Diagram of Hierachical Simulation. CentralController constructs and manages a set of Shuttle agents and the ASRSNetwork. The network provides pathfinding and zone/node resources (cross-aisle, aisles, input/output, CASafe). Shuttle tracks KPIs (travel/wait times, utilization, distances), holds zone locks, and implements deadlock recovery. ModuleFunctions contains plotting/utility helpers. Solid arrows denote calls/ownership; dashed links indicate helper use.

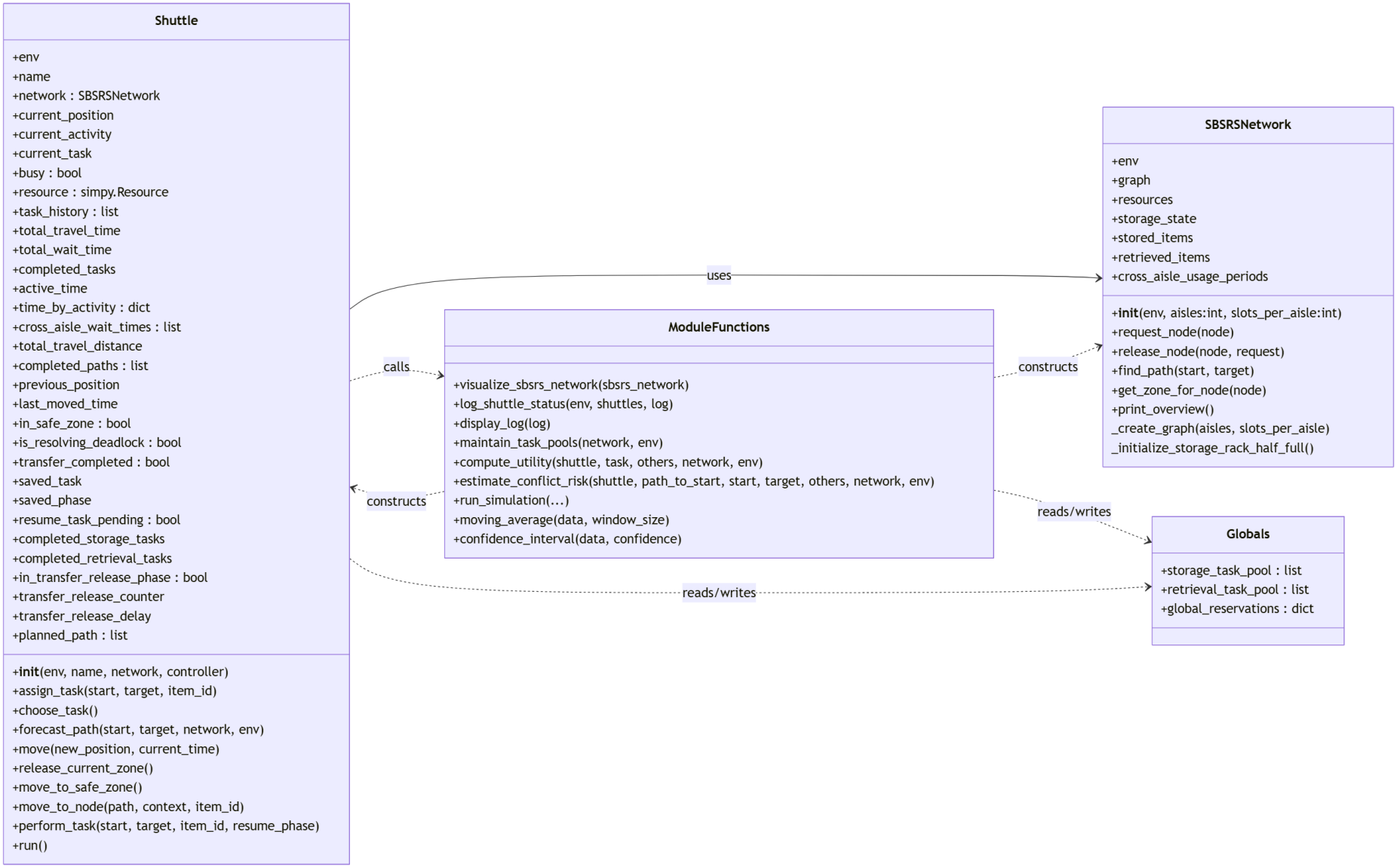


Figure D.4: Class Diagram of Hybrid Simulation. Each Shuttle interacts directly with SBSRSNetwork for paths and locks and consults ModuleFunctions to compute_utility, estimate_conflict_risk, and maintain task pools. Shared Globals hold storage_task_pool, retrieval_task_pool, and global_reservations used for decentralized coordination. The diagram highlights where shuttles read/write shared pools versus merely calling utilities.

E

Verification

Appendix E documents the detailed verification procedures applied to ensure correctness of the simulation model.

E.1. Deterministic Runs

The following aspects were verified across 10 deterministic simulation runs (300 time steps each):

- The shuttle system maintained uniform node path sequences for matching tasks throughout all seed simulations. The planned and executed paths remained identical throughout the simulation. This verifies the state transition accuracy for each shuttle agent starting from Idle and moving through Traveling to Item Transfer.
- The controller used FIFO logic to assign storage and retrieval tasks which executed in the same time slots throughout all simulation runs.
- The model maintained temporal dependencies because tasks became available only after a shuttle enters its idle state were detected and item transfers required node access authorization.
- The shuttle status log produced identical traces for all deterministic runs while showing no variation in the following metrics: completed task counts, task assignment timestamps, path traversal steps and node visits.
- The conflict-resolution logic operates correctly when shuttles enter the same zone (model 1: Hierarchical architecture) or forecast the a conflict on their paths (model 2: Hybrid architceture).
- The system correctly handles deadlock scenarios by observing shuttles move to safe zones before they resume their tasks properly.

As example, the simulation trace logs showed that a shuttle beginning in the `CA-Safe` zone properly yielded to a shuttle exiting from `Input` because of the deadlock resolution policy's priority rules. The simulation trace logs were manually checked for activity transitions which verified the statechart logic. Next to that, the simulation trace logs contained all internal counters including task assignment, task completion and buffer occupancy which were frozen and exported at predetermined time points. The output consistency was verified by comparing snapshots taken from different runs. The model showed no internal variation between runs under deterministic conditions which proved its internal determinism when expected.

The verification method follows the best practices outlined by Law and Kelton [48] who recommend deterministic testing as a first step before adding randomness or performing performance analysis. The model meets the event-driven correctness requirements for reproducibility and causality as specified by Fishman [25] and Law and Kelton [48] and Sargent [70].

E.2. Verification Deadlock Handling

The tests to verify the deadlock behaviour are as follows:

1. Forced Conflict Injection: The model included a test case to introduce conflicts artificially by making two shuttles perform storage duties at locations where their paths intersected at a shared cross-aisle. The shuttle with the lower priority is expected to yield for one timeout or to resolve deadlock. The shuttle enters the Resolving Deadlock state, then moves to the CA-Safe area and then re-enters the system. When analysing the simulation trace, one sees that conflicting shuttles go through their states as expected.

2. CA-Safe Zone Utilization: The entry and exit process to and from the `CA-Safe` zone operations are tracked. During the test period, each shuttle that enters into the safe zone also successfully continues its stored task. This confirms the correct implementation of the recovery logic.

3. Distance-based Priority Rule: The conflicting shuttles and their distances are documented from each deadlock occurrence. The priority rule is as follows:

$$\text{Priority}(A) > \text{Priority}(B) \iff \text{Distance}(A) > \text{Distance}(B)$$

   The shuttle which stands at the greatest distance from the central CA-Safe zone gets priority to move when two shuttles predict a spatial conflict. The heuristic follows the principle that the agent closest to the CA-Safe could be in the path between the conflicting shuttle and the CA-Safe so they should resolve the deadlock. The distance-based rule proved correct in all recorded conflict situations because the shuttle closer to the central CA-Safe zone either yielded with a one time step timeout to the shuttle farther away or started deadlock resolution.

4. Resumption Consistency: The system verified that after exiting the safe zone each shuttle resumed its task from the same phase where it was suspended either at `Seize`, `Store` or `Retrieve`. The system verified the stored task parameters by matching them against the execution logs.

5. Transfer Node Locking: Shuttles need to keep locks on their last nodes of their path until they move at least one step into a new task path. This prevents logical overlaps where another shuttle attempts to seize the same transfer zone before the previous one has fully departed. Especially in the one time step when a shuttle is idle between item transfer and new task start. In the full simulation trace, the release condition “release transfer node only after 1 step of new task executed” was satisfied 100% of the time.

All in all, the model demonstrates strong deadlock handling capabilities because it follows the system design specifications. The study verifies the agent interaction rules through controlled experiments are in alignment with the recommendations presented by Robinson [65].

F

Internal Architecture Behavior Results

Appendix F reports detailed internal performance results for both hierarchical and hybrid control architectures across multiple scenarios.

F.1. Internal Behavior Hierarchical Architecture

The operational performance of the hierarchical control architecture was evaluated across a range of warehouse sizes and shuttle densities. The results clearly show the scalability limits and efficiency trends of this architecture.

System Throughput and Scalability

As shown in Figure F.1, system throughput under hierarchical control decreases noticeably as the warehouse size increases. With 2 shuttles, throughput drops from approximately 163 tasks/hour in a 10×10 system to 82 tasks/hour in a 25×25 system, a reduction of almost 50%. The same trend appears at higher shuttle densities. With 6 shuttles, throughput drops from 117 tasks/hour to 58 tasks/hour across the same size range. This downward trend is visible across all shuttle counts.

Adding more shuttles does decreases throughput, the gains quickly become smaller as the system grows. For example, adding four shuttles (from 2 to 6) decreases throughput by 46 tasks/hour in a 10×10 layout, but only 24 tasks/hour in a 25×25 layout. This diminishing return is most likely caused by increased resource contention and coordination overhead in larger and more complex systems.

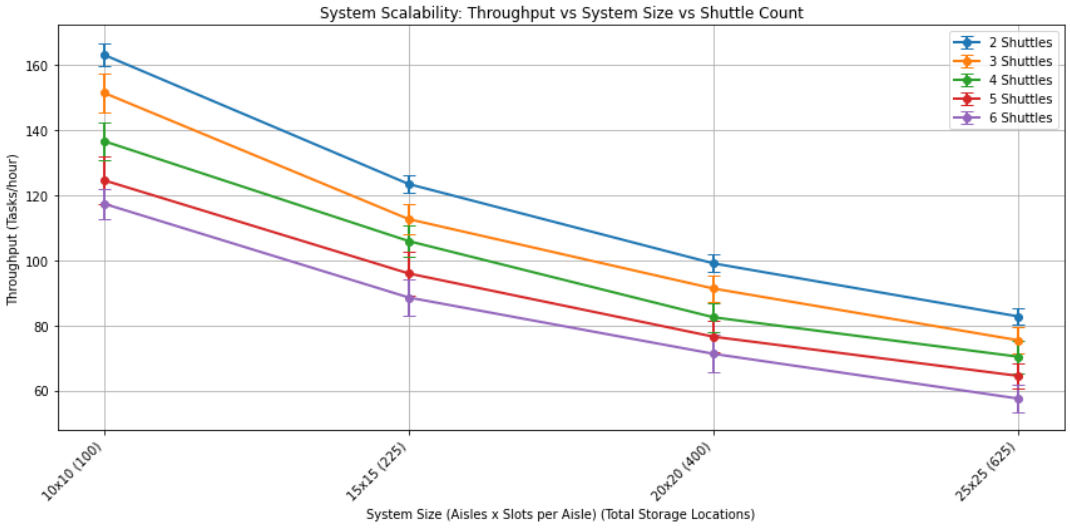


Figure F.1: System Throughput vs. System Size and Shuttle Count (Hierarchical Control)

Normalized Throughput per Shuttle
As shown in Figure F.2, throughput per shuttle also declines as system size and shuttle count increase. In a 10×10 system, throughput per shuttle decreases from 81 tasks/hour/shuttle (2 shuttles) to 20 tasks/hour/shuttle (6 shuttles). In the largest 25×25 system, efficiency drops to 41 tasks/hour/shuttle (2 shuttles) and only 10 tasks/hour/shuttle (6 shuttles), indicating a strong reduction in shuttle utilization. This suggests that the hierarchical architecture struggles to coordinate shuttles efficiently in larger systems, leading to more idle time and congestion.

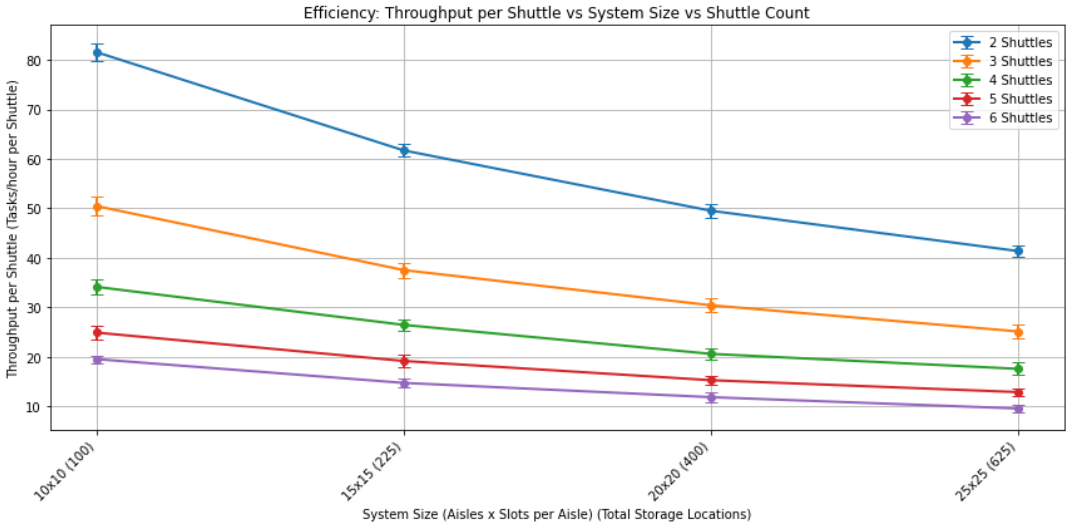


Figure F.2: System Throughput per Shuttle vs. System Size and Shuttle Count (Hierarchical Control).

Heatmap Analysis: Throughput and Normalized Throughput
The heatmaps in Figure F.3 confirms the trends. The throughput heatmap shows that the highest performance occurs in small, low-density systems. As warehouse size or shuttle count increases, throughput values drop sharply, reaching a minimum of around 58 tasks/hour in the largest and most densely populated configuration. This is likely caused by a combination of longer travel distances.

The normalized throughput heatmap, shown in Figure F.4, indicates that operational efficiency falls from 7.38 tasks/hour per unit distance (10×10, 2 shuttles) to 1.03 (25×25, 6 shuttles), which is a reduction of more than 85%. This is likely caused by the limited ability of the central controller to handle congestion as shuttle density increases.

Summary Hierarchical Architecture Performance
In summary, the hierarchical control architecture performs best in small, lightly loaded systems. As system size and shuttle density increase, both throughput and operational efficiency decline sharply, and the marginal utility of additional shuttles is significantly reduced. These quantitative results indicate that the scalability of hierarchical control is likely limited by its sequential coordination policies and increasing resource contention at higher system complexities.

- System throughput declines sharply with increasing system size, dropping from approximately 225 tasks/hour in a 5×10 layout to just 86 tasks/hour in a 25×25 layout, a 62% decrease.
- Efficiency loss is also evident: throughput per shuttle for 2 shuttles falls from about 108 to 39 tasks/hour/shuttle (a 64% reduction), while for 4 shuttles, efficiency drops from 56 to 22 tasks/hour/shuttle as system size increases (a 61% decrease).
- Adding more shuttles leads to performance saturation: in large systems (25×25), throughput for 4, 5, and 6 shuttles plateaus at around 85-87 tasks/hour, indicating that additional shuttles do not yield significant performance gains.
- Normalized throughput (efficiency per travel distance) declines from 8.1 to 1.1 tasks/hour/unit distance as the system scales from 5×10 with 2 shuttles to 25×25 with 6 shuttles, representing an 86% reduction in operational efficiency.

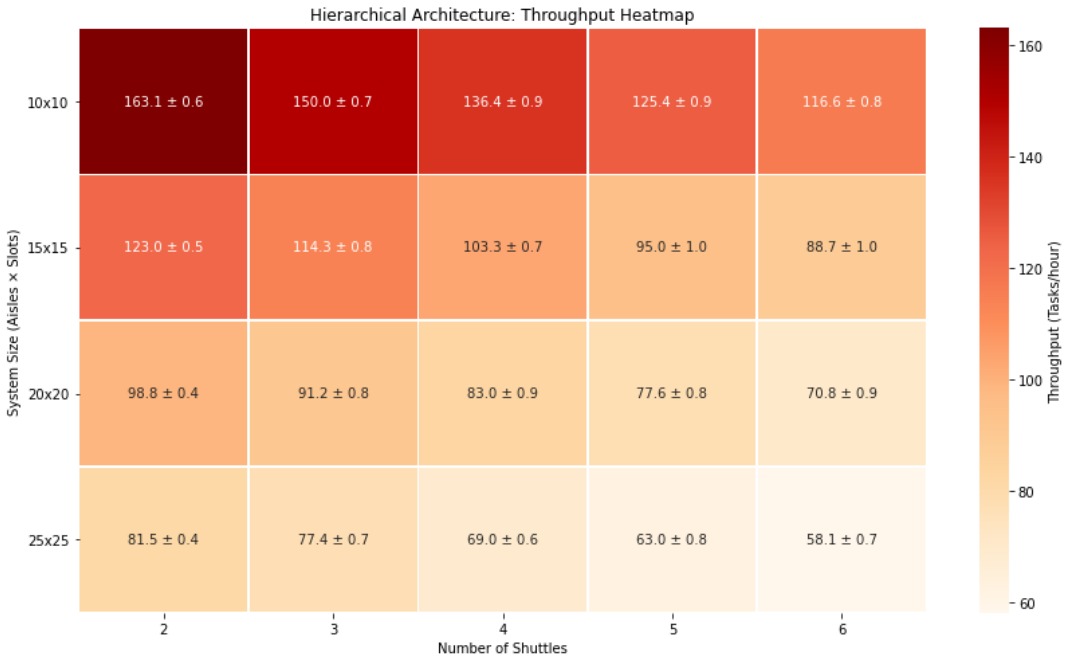


Figure F.3: Heatmap of System Throughput Across System Sizes and Shuttle Counts (Hierarchical Control).

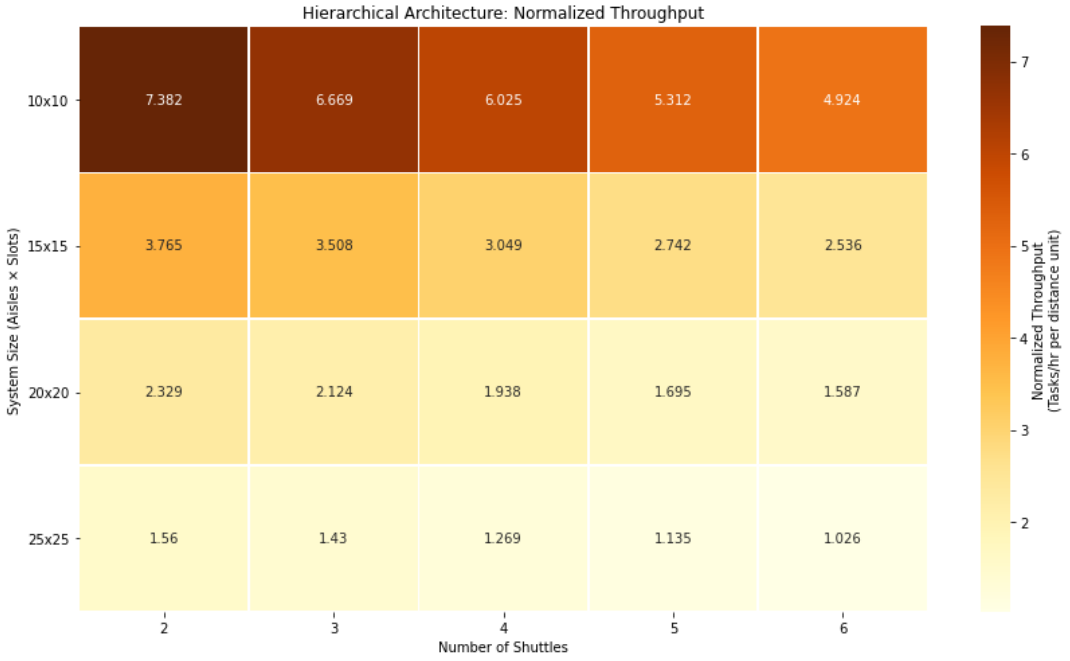


Figure F.4: Heatmap of Normalized Throughput per Unit Travel Distance (Hierarchical Control)

- Layout geometry further impacts performance: for systems with equal storage capacity, a 5×20 layout (long cross-aisle) achieves only 184 tasks/hour, compared to 225 tasks/hour in a 10×10 layout (short cross-aisle), an ̃18% throughput loss likely attributable to increased cross-aisle traversal distances.

F.2. Internal Behavior Hybrid Architecture

The performance of the hybrid control architecture was evaluated using the same set of scenarios, so its scalability, efficiency, and utilization can be directly compared to the hierarchical model.

System Throughput and Scalability

As illustrated in Figure F.5, the hybrid architecture maintains relatively high throughput across all system sizes and shuttle counts. With 2 shuttles, throughput decreases from approximately 187 tasks/hour in a 10×10 system to 100 tasks/hour in a 25×25 system (a drop of about 47%), which is notably less than the decrease observed under the hierarchical architecture. More importantly, throughput in the hybrid model continues to increase, or stays within a small range, as more shuttles are added. For example, in the 25×25 layout, throughput rises from 100 tasks/hour (2 shuttles) to 111 tasks/hour (6 shuttles), indicating that the system is still able to benefit from additional resources.

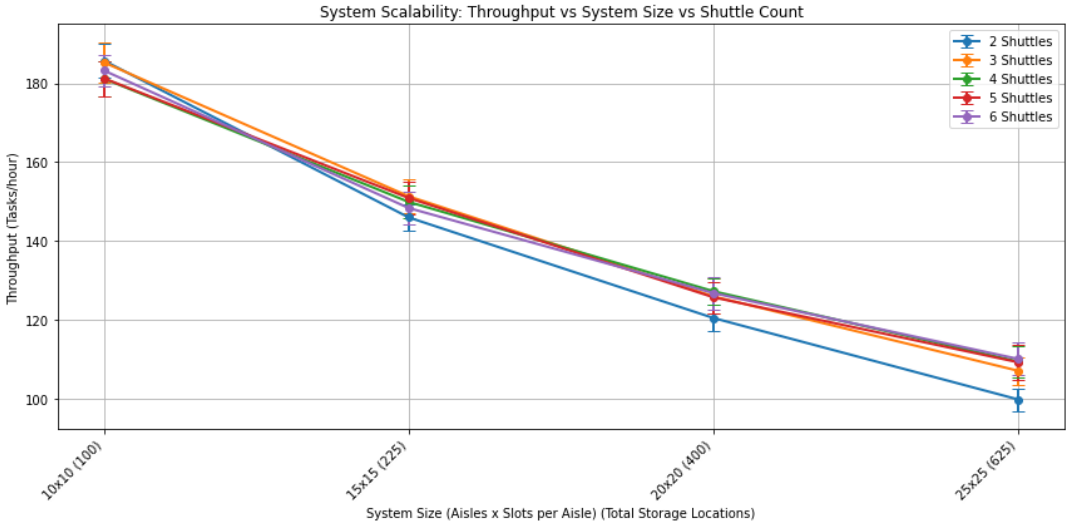


Figure F.5: System Throughput vs. System Size and Shuttle Count (Hybrid Control)

Normalized Throughput per Shuttle

Figure F.6 shows that throughput per shuttle does decrease as more shuttles are added, but the drop is less severe compared to the hierarchical case. In a 10×10 system, throughput per shuttle falls from 93 tasks/hour/shuttle (2 shuttles) to 31 tasks/hour/shuttle (6 shuttles), and in a 25×25 system from 50 to 18 tasks/hour/shuttle. This suggests that the hybrid model makes better use of additional shuttles, particularly in larger and more densely populated systems.

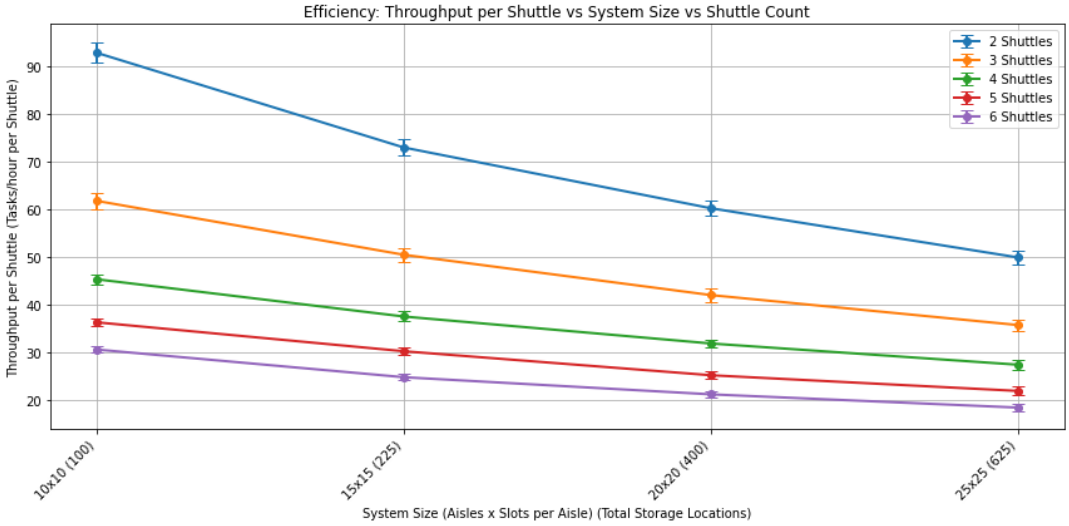


Figure F.6: System Efficiency: System Throughput per Shuttle vs. System Size and Shuttle Count (Hybrid Control).

Heatmap Analysis: Throughput and Normalized Throughput

The heatmaps, Figure F.7 and Figure F.8, show that the hybrid architecture maintains high throughput and operational efficiency across all tested scenarios. Throughput values remain above 99 tasks/hour even in the most challenging configurations and stay above 180 tasks/hour in smaller systems. Although normalized throughput decreases as system size and shuttle count increase, it is still higher than in the hierarchical architecture. For example, normalized throughput drops from 8.6 to 1.7 tasks/hour per unit distance when moving from a 10×10 system with 2 shuttles to a 25×25 system with 6 shuttles (≈ 80% decrease).

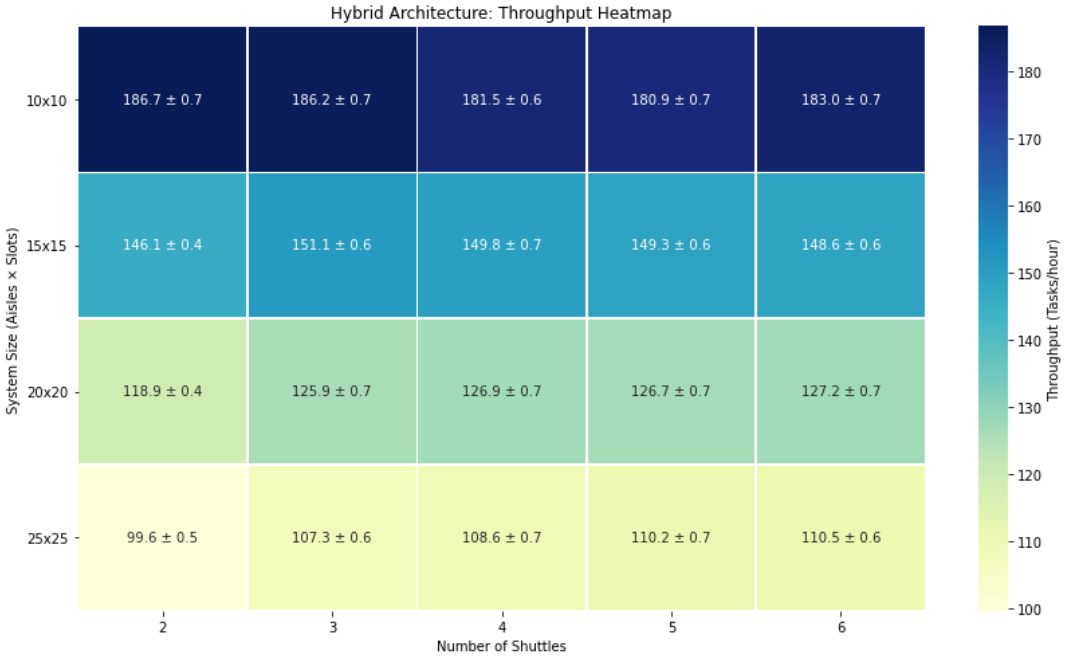


Figure F.7: Heatmap of System Throughput Across System Sizes and Shuttle Counts (Hybrid Control)

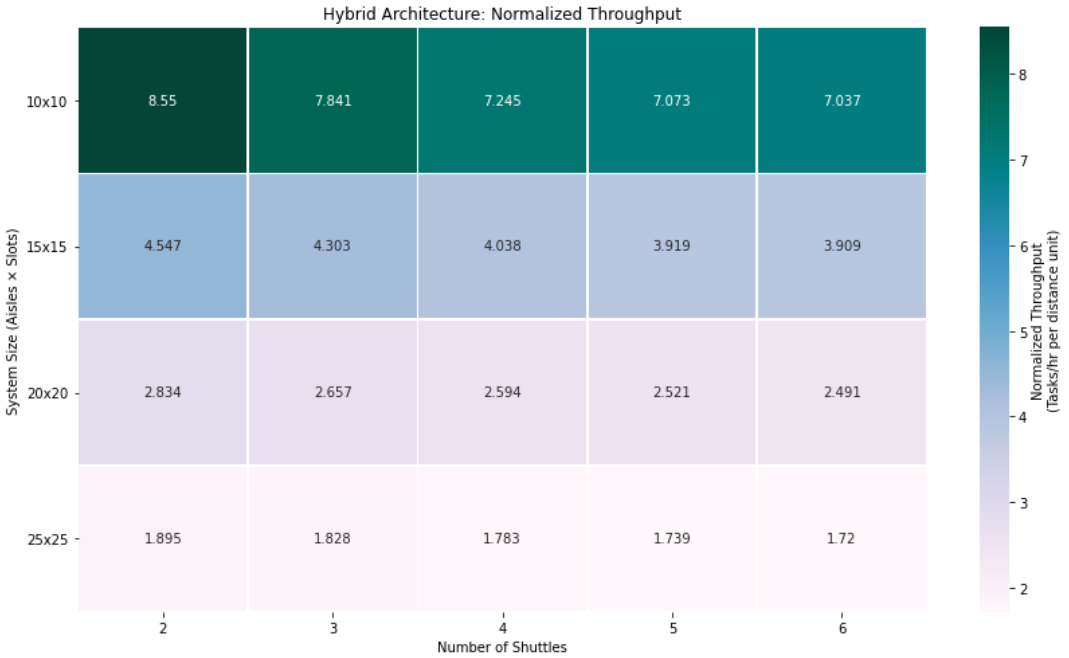


Figure F.8: Heatmap of Normalized Throughput per Unit Travel Distance (Hybrid Control)

An interesting observation of the hybrid architecture is that it supports the addition of more shuttles without decreasing in throughput. Across all evaluated layouts, each additional shuttle continued to improve throughput or reached a throughput plateau but never significantly decreases. This indicates that distributed decision-making and adaptive routing help to reduce resource contention and prevent the performance saturation seen in more centralized systems.

Summary of Hybrid Architecture Performance
Overall, the hybrid control architecture shows strong scalability and good shuttle utilization. It keeps throughput and operational efficiency high across a wide range of system sizes and shuttle counts. Because of its local control, the system can still benefit from additional shuttles even in large, dense environments. These results suggest that hybrid control is well suited to dynamic, high-capacity SBSRS and aligns with the flexibility and adaptability required for Industry 4.0 applications.

- System throughput remains high, ranging from 187 tasks/hour (10×10, 2 shuttles) to 111 tasks/hour (25×25, 6 shuttles)
- Throughput per shuttle decreases with increasing shuttle count but stays above 18 tasks/hour/shuttle in the largest system
- Normalized throughput drops from 8.6 to 1.7 tasks/hour per unit distance (10×10, 2 shuttles → 25×25, 6 shuttles)
- Adding more shuttles consistently increases throughput across all scenarios (no performance saturation)
- Decentralised control enables sustained scalability and efficient use of resources